Hybrid Mapping for Static and Non-static Indoor Environments

by

Clara Gómez Blázquez

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in

Electrical Engineering, Electronics and Automation

University Carlos III of Madrid

Advisor:
Dr. Ramón Barber Castaño

Tutor:
Dr. Ramón Barber Castaño

March 2021

To the past, my childhood,
to the present, my family,
and to the future.

Acknowledgements

The kind support and help of many people made this thesis possible. I would like to take this space to thank them as there is a bit of each of them in this document.

First of all, I would like to thank my thesis advisor, Ramón Barber, for his helpful guidance during the first years of the thesis and the confidence and freedom during the subsequent years. He allowed me to drive this thesis and change its aims as my research interests changed along these years. I want to thank also Luis Moreno, who taught me the importance of solid mathematical backgrounds, for the fruitful and enriching discussions.

Secondly, I would like to thank my colleagues from the research group and the RoboticsLab. It has been great working with you, both in research and teaching. Alejandra, Raúl, Edwin y Juanmi habéis sido un gran apoyo. Me gustaría agerdecer especialmente a Alejandra, mi gran compañera de doctorado y amiga. Me siento afortunada de haber trabajado contigo estos años, codo con codo, hemos aprendido mucho juntas y no sé que habría sido de mi tesis si no hubiese estado trabajando contigo. I would also like to thank Erik, who came to the lab as a research visitor and ended up being a great compañero y amigo. To both of you, I hope this journey does not end here and we continue working together in the future.

A big part of this thesis has been developed abroad, making it a very enriching experience both professionally and personally. I had the chance to collaborate, discuss and learn from many researchers during these years. I would like to thank the ASL team of ETH Zurich, specially Roland Siegwart for welcoming me to his lab, Juan Nieto for his guidance during the months I worked there and Cesar Cadena, Marius Fehr and Alex Millane for the interesting discussions and help. Thanks also to Nancy Amato, who was also visiting the lab during those months and I had the chance to discuss some ideas with her and learn from her. I want to thank also Robert Babuška for welcoming us at CIIRC in Prague and Erik for making that research visit possible and being such a great host. I would also like to thank Cyrill Stachniss for his invaluable guidance in our weekly virtual meetings during the last year and for teaching me about probabilistic robotics and localization.

Me gustaría agradecer también a las personas que me han ayudado a revisar este documento: Alejandra, Julia, Jesús, papá, Javi y Juan.

Por último, me gustaría dedicar unas palabras para agredecer a las personas más importantes de mi vida. Mis amigas y amigos se merecerían un parrafo para cada uno de ellos. Bea, Maria Victoria, Julia, Eva, Marcos, Pablo, David, Lucas, Pedro y Gochi: sois geniales y me habéis ayudado a desconectar y disfrutar de estos años con nuestras risas e historias. Me gustaría agredecer a Jesús por que esta tesis ha sido posible a tu lado (en el último año literalmente). Gracias por tu apoyo y ayuda siempre que la he necesitado. Podría escribir un libro entero agradeciendo a mi familia su apoyo y todo lo que me han dado. Papá, mamá, Javi, Juan, Aratxa, Gael, Jesús, tito, tita y Mariuca, sois lo mejor que tengo. Mis padres, que con su ejemplo me han inculcado lo que significa el esfuerzo y luchar por lo que quieres y que me han apoyado, comprendido y querido como nadie. Mis hermanos, que son mis compañeros de vida, Javi, mostrándome siempre el camino y, Juan, enseñándome que los caminos son infinitos. Vuestro apoyo y cariño son parte de estos años de duro trabajo. Y Gael, el pequeño de la familia, verte aprender y descubrir el mundo ha sido muy estimulante para mi propio aprendizaje como *doctora de robots*. Me has enseñado nuevas formas de querer y ser querida. Tú eres el futuro.

Published Content and Contributions

Parts of this thesis are based on work published in the following journal and conference papers and book chapters:

Journal Publications (3)

Gomez, C., Hernandez, A. C., Derner, E., Barber, R. and Babuška, R. (2020). *Object-Based Pose Graph for Dynamic Indoor Environments.* IEEE Robotics and Automation Letters (**IROS+RA-L submission**), vol. 5, no. 4, pp. 5401-5408. DOI: 10.1109/LRA.2020.3007402.

- This journal is included in this thesis in Chapter 6. The material from this source is not singled out.

Gomez, C., Hernandez, A. C. and Barber, R. (2019). *Topological Frontier-Based Exploration and Map-Building Using Semantic Information.* **Sensors**, vol. 19, no. 20, p. 4595. DOI: 10.3390/s19204595.

- This journal is included in this thesis in Chapter 5. The material from this source is not singled out.

Gomez, C., Hernandez, A. C., Crespo, J. and Barber, R. (2017). *A Topological Navigation System for Indoor Environments based on Perception Events.* International Journal of Advanced Robotic Systems (IJARS), vol. 14, no. 1. DOI: 10.1177/1729881416678134.

- This journal is partially included in this thesis in Chapter 4. The material from this source is not singled out.

Conference Publications (7)

Gomez, C., Hernandez, A. C., Barber, R. and Stachniss, C. (2021). *Localization Exploiting Semantic and Metric Information in Non-static Indoor Environments.* IEEE/RSJ International Conference on Intelligent Robots and Systems (**IROS**). [sent, pending notification]

- This paper is included in this thesis in Chapter 7. The material from this source is not singled out.

Gomez, C., Marius, F., Millane, A., Hernandez, A. C., Nieto, J., Barber, R. and Siegwart, R. (2020). *Hybrid Topological and 3D Dense Mapping through Autonomous Exploration for Large Indoor Environments.* IEEE International Conference on Robotics and Automation (**ICRA**), pp. 9673-9679. DOI: 10.1109/ICRA40945.2020.9197226.

- This paper is included in this thesis in Chapters 3 , 4 and 7. The material from this source is not singled out.

Gomez, C., Hernandez, A. C., Derner, E. and Barber, R. (2019) *Semantic Localization through Propagation of Scene Information in a Hierarchical Model.* IEEE European Conference on Mobile Robots (ECMR), pp. 1-6. DOI: 10.1109/ECMR.2019.8870972.

- This paper is included in this thesis in Chapter 7. The material from this source is not singled out.

Gomez, C., Hernandez, A., Barber, R., Moreno, L. and Martinez-Mozos, O. (2019). *Localization of Mobile Robots Incorporating Scene Information in a Hierarchical Model.* IEEE International Conference on Robotic Computing (IRC), pp. 429-430. DOI: 10.1109/IRC.2019.00084.

- This paper is included in this thesis in Chapter 7. The material from this source is not singled out.

Hernandez, A. C., **Gomez, C.** and Barber, R. (2019). MiNERVA: Toposemantic Navigation Model based on Visual Information for Indoor Enviroments. IFAC-PapersOnLine (Symposium on Intelligent Autonomous Vehicles (IAV)), vol. 52, no. 8, pp. 43-48. DOI: 10.1016/j.ifacol.2019.08.046.

- This paper is partially included in this thesis in Chapters 3 and 4. The material from this source is not singled out.

Gomez, C., Hernandez, A. C., Moreno, L. and Barber, R. (2018). *Qualitative Geometrical Uncertainty in a Topological Robotic Localization System.* IEEE International Conference on Control, Artificial Intelligence, Robotics and Optimization (ICCAIRO), pp. 183-188. DOI: 10.1109/ICCAIRO.2018.00038.

- This paper is partially included in this thesis in Chapter 7. The material from this source is referenced in the text.

Gomez, C., Hernandez, A. C., Crespo, J. and Barber, R. (2017). *Uncertainty-based Localization in a Topological Robotic Navigation System.* IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 67-72. DOI: 10.1109/ICARSC.2017.7964054.

- This paper is included in this thesis in Chapter 7. The material from this source is not singled out.

Book Chapters (1)

Barber, R., Crespo, J., **Gomez, C.**, Hernandez, A. C., and Galli, M. (2018). *Mobile Robot Navigation in Indoor Environments: Geometric, Topological, and Semantic Navigation.* Applications of Mobile Robots, IntechOpen. DOI: 10.5772/intechopen.79842.

- This book chapter is partially included in this thesis in Chapters 3 and 4. The material from this source is not singled out.

Although not included in this thesis, the work presented here has contributed to other published works:

Other Journal and Conference Publications (9)

Derner, E., **Gomez, C.**, Hernandez, A. C., Barber, R., and Babuška, R. (2021). *Change detection using weighted features for image-based localization.* Robotics and Autonomous Systems (**RAS**), vol. 135, p. 103676. DOI: 10.1016/j.robot.2020.103676.

Hernandez, A. C., Derner, E., **Gomez, C.**, Barber, R., and Babuška, R. (2020). *Efficient Object Search Through Probability-Based Viewpoint Selection.* IEEE/RSJ International Conference on Intelligent Robots and Systems (**IROS**), pp. 6172–6179. DOI: 10.1109/IROS45743.2020.9340989.

Derner, E., **Gomez, C.**, Hernandez, A. C., Barber, R., and Babuška, R. (2019). *Towards Life-Long Autonomy of Mobile Robots Through Feature-Based Change Detection.* IEEE European Conference on Mobile Robots (ECMR), pp. 1–6. DOI: 10.1109/ECMR.2019.8870940.

Hernandez, A. C., **Gomez, C.**, Derner, E., and Barber, R. (2019). *Indoor Scene Recognition Based on Weighted Voting Schemes.* IEEE European Conference on Mobile Robots (ECMR), pp. 1-6. DOI: 10.1109/ECMR.2019.8870931.

Hernandez, A. C., **Gomez, C.**, Martinez-Mozos, O. and Barber, R. (2018). *Object-based Probabilistic Place Recognition for Indoor Human Environments.* IEEE International Conference on Control, Artificial Intelligence, Robotics and Optimization (ICCAIRO), pp. 177-182. DOI: 10.1109/ICCAIRO.2018.00037.

Crespo, J., **Gomez, C.**, Hernandez, A. C. and Barber, R. (2017). *A Semantic Labelling of the Environment based on What People Do.* **Sensors**, vol. 17, no 2, p. 260. DOI: 10.3390/s17020260.

Hernandez, A. C., **Gomez, C.**, Crespo, J. and Barber, R. (2017). *Adding Uncertainty to an Object Detection System for Mobile Robot Navigation.* IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT), pp. 7-12. DOI: 10.1109/SMC-IT.2017.9.

Hernandez, A. C., **Gomez, C.**, Crespo, J. and Barber, R. (2016). *Object Detection Applied to Indoor Environments for Mobile Robot Navigation.* **Sensors**, vol. 16, no 8, p. 1180. DOI: 10.3390/s16081180.

Hernandez, A. C., **Gomez, C.**, Crespo, J. and Barber, R. (2016). *Object Classification in Natural Environments for Mobile Robot Navigation.* IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 217-222. 10.1109/ICARSC.2016.55.

Other Research Merits

The works presented in this thesis have been partially developed in collaboration with other institutions apart from our lab at University Carlos III of Madrid. I have been pleased to collaborate with colleagues from:

- Autonomous System Lab (ASL) in ETH Zurich. The head of the lab, Roland Siegwart, welcomed me for a 3-month research visit that finished as a fruitful collaboration and conference publication of the work *Hybrid Topological and 3D Dense Mapping through Autonomous Exploration for Large Indoor Environments*.
- Czech Institute of Informatics, Robotics and Cybernetics (CIIRC) in CTU Prague. A second research visit during my PhD was possible thanks to Robert Babuška that invited me to collaborate in his lab in a 2-month research visit. This collaboration resulted in several conference and journal publications, being relevant for this thesis the journal publication and conference presentation titled *Object-Based Pose Graph for Dynamic Indoor Environments*.
- Photogrammetry and Robotics Lab in University of Bonn. Firstly, Cyrill Stachniss invited me to a 3-month research visit in his lab and an in-person collaboration. Due to the pandemic caused by COVID-19, the research visit was not possible but we successfully held an online collaboration whose expected outcome is a conference publication (already under revision).

Abstract

Indoor environments populated by humans, such as houses, offices or universities, involve a great complexity due to the diversity of geometries and situations that they may present. Apart from the size of the environment, they can contain multiple rooms distributed into floors and corridors, repetitive structures and loops, and they can get as complicated as one can imagine. In addition, the structure and situations that the environment present may vary over time as objects could be moved, doors can be frequently opened or closed and places can be used for different purposes. Mobile robots need to solve these challenging situations in order to successfully operate in the environment. The main tools that a mobile robot has for dealing with these situations relate to navigation and perception and comprise mapping, localization, path planning and map adaptation. In this thesis, we try to address some of the open problems in robot navigation in non-static indoor environments. We focus on house-like environments as the work is framed into the HEROITEA research project that aims attention at helping elderly people with their everyday-life activities at their homes. This thesis contributes to HEROITEA with a complete robotic mapping system and map adaptation that grants safe navigation and understanding of the environment. Moreover, we provide localization and path planning strategies within the resulting map to further operate in the environment.

The first problem tackled in this thesis is robot mapping in static indoor environments. We propose a hybrid mapping method that structures the information gathered from the environment into several maps. The hybrid map contains diverse knowledge of the environment such as its structure, the navigable and blocked paths, and semantic knowledge, such as the objects or scenes in the environment. All this information is separated into different components of the hybrid map that are interconnected so the system can, at any time, benefit from the information contained in every component. In addition to the conceptual conception of the hybrid map, we have also developed building procedures and an exploration algorithm to autonomous build the hybrid map.

However, indoor environments populated by humans are far from being static as the environment may change over time. For this reason, the second problem tackled in this thesis is the adaptation of the map to non-static environments. We propose an object-based probabilistic map adaptation that calculates the likelihood of moving or remaining in its place for the different objects in the environment.

Finally, a map is just a description of the environment whose importance is mostly related to how the map is used. In addition, map representations are more valuable as long as they offer a wider range of applications. Therefore, the third problem that we approach in this thesis is exploiting the intrinsic characteristics of the hybrid map in order to enhance the performance of localization and path planning methods. The particular objectives of these approaches are precision for robot localization and efficiency for path planning in terms of execution time and traveled distance.

We evaluate our proposed methods in a diversity of simulated and real-world indoor environments. In this extensive evaluation, we show that hybrid maps can be efficiently built and maintained over time and they open up for new possibilities for localization and path planning. In this thesis, we show an increase in localization precision and robustness and an improvement in path planning performance.

In sum, this thesis makes several contributions in the context of robot navigation in indoor environments, and especially in hybrid mapping. Hybrid maps offer higher efficiency during map building and other applications such as localization and path planning. In addition, we highlight the necessity of dealing with the dynamics of indoor environments and the benefits of combining topological, semantic and metric information to the autonomy of a mobile robot.

Resumen

Los entornos de interiores habitados por personas, como casas, oficinas o universidades, entrañan una gran complejidad por la diversidad de geometrías y situaciones que pueden ocurrir. Aparte de las diferencias en tamaño, estos entornos pueden contener muchas habitaciones organizadas en diferentes plantas o pasillos, pueden presentar estructuras repetitivas o bucles de tal forma que los entornos pueden llegar a ser tan complejos como uno se pueda imaginar. Además, la estructura y el estado del entorno pueden variar con el tiempo, ya que los objetos pueden moverse, las puertas pueden estar cerradas o abiertas y diferentes espacios pueden ser usados para diferentes propósitos. Los robots móviles necesitan resolver estas situaciones difíciles para poder funcionar de una forma satisfactoria. Las principales herramientas que tiene un robot móvil para manejar estas situaciones están relacionadas con la navegación y la percepción y comprenden el mapeado, la localización, la planificación de trayectorias y la adaptación del mapa. En esta tesis, abordamos algunos de los problemas sin resolver de la navegación de robots móviles en entornos de interiores no estáticos. Nos centramos en entornos tipo casa ya que este trabajo se enmarca en el proyecto de investigación HEROITEA que se enfoca en ayudar a personas ancianas en tareas cotidianas del hogar. Esta tesis contribuye al proyecto HEROITEA con un sistema completo de mapeado y adaptación del mapa que asegura una navegación segura y la comprensión del entorno. Además, aportamos métodos de localización y planificación de trayectorias usando el mapa construido para realizar nuevas tareas en el entorno.

El primer problema que se aborda en esta tesis es el mapeado de entornos de interiores estáticos por parte de un robot. Proponemos un método de mapeado híbrido que estructura la información capturada en varios mapas. El mapa híbrido contiene información sobre la estructura del entorno, las trayectorias libres y bloqueadas y también incluye información semántica, como los objetos y escenas en el entorno. Toda esta información está separada en diferentes componentes del mapa híbrido que están interconectados de tal forma que el sistema puede beneficiarse en cualquier momento de la información contenida en cada componente. Además de la definición conceptual del mapa híbrido, hemos desarrollado unos procedimientos para construir el mapa y un algoritmo de exploración que permite que esta construcción se realice autónomamente.

Sin embargo, los entornos de interiores habitados por personas están lejos de ser estáticos ya que pueden cambiar a lo largo del tiempo. Por esta razón, el segundo problema que intentamos solucionar en esta tesis es la adaptación del mapa para entornos no estáticos. Proponemos un método probabilístico de adaptación del mapa basado en objetos que calcula la probabilidad de que cada objeto en el entorno haya sido movido o permanezca en su posición anterior.

Para terminar, un mapa es simplemente una descripción del entorno cuya importancia está principalmente relacionada con su uso. Por ello, los mapas más valiosos serán los que ofrezcan un rango mayor de aplicaciones. Para abordar este asunto, el tercer problema que intentamos solucionar es explotar las características intrínsecas del mapa híbrido para mejorar el desempeño de métodos de localización y de planificación de trayectorias usando el mapa híbrido. El objetivo principal de estos métodos es aumentar la precisión en la localización del robot y la eficiencia en la planificación de trayectorias en relación al tiempo de ejecución y la distancia recorrida.

Hemos evaluado los métodos propuestos en una variedad de entornos de interiores simulados y reales. En esta extensa evaluación, mostramos que los mapas híbridos pueden construirse y mantenerse en el tiempo de forma eficiente y que dan lugar a nuevas posibilidades en cuanto a localización y planificación de trayectorias. En esta tesis, mostramos un aumento en la precisión y robustez en la localización y una mejora en el desempeño de la planificación de trayectorias.

En resumen, esta tesis lleva a cabo diversas contribuciones en el ámbito de la navegación de robots móviles en entornos de interiores, y especialmente en mapeado híbrido. Los mapas híbridos ofrecen más eficiencia durante la construcción del mapa y en otras tareas como la localización y la planificación de trayectorias. Además, resaltamos la necesidad de tratar los cambios en entornos de interiores y los beneficios de combinar información topológica, semántica y métrica para la autonomía del robot.

Contents

List of Figures

List of Tables

1

Introduction

Robot navigation and mapping serves to provide a robot with navigation capabilities to move safely in the environment and take intelligent decisions regarding how and where to move for a certain task. Robots can help others or develop tasks in indoor or outdoor environments and in the company of humans or not. Thus, the complexity of robot navigation tasks vary according to the size of the working environment of the robot, its structureness, the influence of humans and the changeability of the environment, among others. Indoor environments, such as houses, hospitals, office buildings or universities, present a high complexity since they contain a great variety of geometric structures and situations. These environments can be very complex and can contain hundreds of rooms distributed into multiple floors, areas and corridors and can present a great diversity of connections between rooms. In addition to the geometric variety of indoor environments, they may contain a diversity of situations that change over time, including densely-crowded hallways, corridors with doors frequently opening and closing, cluttered rooms and furniture that occasionally moves. Moreover, the complexity of indoor environments does not reside solely in the diversity of structures and situation, indoor environments are GPS denied and this fact obscures robot operation compared to open outdoor environments. It is essential that mobile robots operating in indoor environments address the scale, complexity and information acquisition of the environments for robustly performing everyday-life tasks.

This thesis is framed into HEROITEA[1] research project that focuses on helping elderly people with their everyday-life activities at their homes. Tasks such as cooking or carrying things can become challenging with the age and HEROITEA tries to give a robotic solution to this problem. For a robot to operate with humans in their environments, different functionalities are required: perception and understanding of objects and scenes, human-robot interaction, mapping and navigation through the environment, etc. This thesis is focused mainly on developing the mapping system for HEROITEA research project and contributing to its navigation tasks. For this reason, the proposed method must deal with dynamic indoor environments, be autonomous in every step of the mapping and navigation processes and capture as much relevant information as possible of the environment.

1.1 Problem Statement and Motivation

Today, many complex tasks are starting to be carried out by mobile robots: spatial exploration robots have been deployed to Mars [106] and the moon [169]; autonomous vehicles are transporting goods in warehouses [24, 44] and cleaning robots are helping people with their household chores [13, 86, 160]; mobile robots are also deployed to help humans in other tasks such as inspecting and patrolling in dangerous environments [185] or taking care of agricultural tasks [72] and livestock [79]. To operate autonomously, all the aforementioned robots and robotic systems need robust and complete navigation modules. These works, and many others, offer interesting solutions to limited scopes of robot navigation in specific domains that will contribute to the achievement of robust long-term robot navigation. However, in spite of their successes, the goal of long-term mapping and navigation is far from being achieved.

Depending on the environment and application, mobile robots need to solve different challenges including localization, mapping, path and action planning, perception, reasoning and much more. In this thesis, we focus on the tasks of mapping, map maintenance, localization and path planning. Mobile robots operating in outdoor environments can be provided with accurate satellite maps and global positioning systems that contribute to the navigation task. However, indoor environments are blind spots for satellites and thus, satellite maps and global positioning are not available.

[1]HEROITEA: Heterogeneous Intelligent Multi-Robot Team for Assistance of Elderly People (RTI2018-095599-B-C21), funded by Spanish Ministerio de Economia y Competitividad

For most indoor environments, map construction has to be performed by the robot and it is a key factor for the success of robot operation. The motivation of this work is to develop a novel complete representation of the environment that captures the precise structure and complexity of the environment in a structured fashion and allows further robot navigation and adaptation to changes in the environment to contribute in filling the gap of long-term mapping and navigation. For this purpose, this thesis introduces a new hybrid map representation that contains useful metric, topological and semantic information for navigation and map maintenance in indoor non-static environments. We complement this mapping method with a map adaptation algorithm based on metric and semantic information to overcome changes caused by movable objects. Moreover, given that indoor environments are GPS denied, we solve localization and path planning tasks relative to the hybrid map representation.

1.2 Objectives

From the challenges proposed in the previous section, the aim of this thesis is to develop a robust autonomous mapping method for non-static indoor environments that enables safe navigation and operation. This aim leads to several specific objectives:

- Defining a model of the environment that allows a map to contain diverse and useful information of the environment, specially of its structure, the elements contained in the environment, the relation between them and the 3D representation of the environment.

- Developing an exploration strategy that allows to autonomously build the map and gather the required information.

- Detecting and managing the changes in the environment in long-term operation and update the map with the learned information.

- Localizing the robot within the hybrid representation through probabilistic methods that manage the uncertainty of the environment using the different sources of information contained in the hybrid map.

- Navigating through the environment using the hybrid map and path planning strategies to produce efficient and safe trajectories.

This thesis will examine previous attempts to achieve the stated objectives and it will give a new solution to them. The proposed solution to each objective will be thoughtfully evaluated and justified through experiments and comparisons.

1.3 Contributions

This work intends to provide new answers to already-known problems that outperform or give a new perspective with regard to current solutions. Our primary contributions to robot mapping and navigation are three-fold: on building the map representation, maintaining the map representation and using the map representation. Research efforts on building the map representation focused on including a wider range of information in a structured fashion and in developing more efficient mapping methods. When maintaining the map representation, we sought for a novel flexible adaptation of the map based on objects. Thirdly, we propose an innovative solution in terms of using the map representation through localization and path planning strategies combining different components of the hybrid map.

With regard to the definition of the model of the environment, a new hybrid mapping method is proposed that includes structural, relational, 3D and object information structured into different layers of the hybrid map. The novel conception of the hybrid map simplifies the construction and abstraction of the map and enlarges the navigation capabilities of the robot compared to current works. In addition, thanks to the proposed method, building the whole hybrid map outperforms standard 3D mapping in terms of memory consumption and the proposed submapping strategy improves upon fixed submapping in terms of memory consumption and number of submaps. Furthermore, a novel exploration algorithm for map construction is proposed. The novelty of the exploration algorithm lies in a cost-utility function that reduces execution time and traveled distance with regard to relevant state-of-the-art methods.

The next contribution of this thesis refers to the adaptation to non-static environments. We provide a novel probabilistic method that learns the movability of the objects in the environment. The changes in the environment are captured and added to the map in a probabilistic fashion that represents how movable or static an object is and can lately be used for determining the most trustful elements. This method goes beyond other proposed methods, such as binary classifications between static and movable elements, and broadens map adaptation applicability.

Thanks to the hybrid definition of the model, novel methods for the usage of the map can be proposed. From a topological localization perspective, we show how the hybrid map can improve the estimation of robot location compared to using the information of just one component of the map. From a path planning perspective, we have designed a novel path planning strategy that, using several components of the map, goes beyond standard path planning techniques in terms of planning and execution time and traveled distance. Our last contribution lies in the usage of semantic and metric information for precise pose estimation in static and non-static environments. The combination of both sources of information provides a more accurate robot pose than just using one of them.

1.4 Structure of the Document

The presented objectives and contributions have been distributed throughout this thesis into eight chapters. Chapter 2 revises the related work in hybrid mapping for mobile robots describing the most relevant works in the topic and positioning this thesis with regard to them. This revision includes a historical compilation of the proposed methods, as well as, a detailed description of the trends and current state of the hybrid mapping paradigm.

Once the related work and the context of this thesis are explained, in Chapter 3, the proposed model of the environment is described. The model of the environment consists of a hybrid map structured in several components that contain different information. The theoretical description and requirements for each component of the map, the contained information and the relation between components are included. This chapter states the main definitions and limitations of the model, whereas the subsequent chapters address specific tasks involving this model.

Chapter 4 presents the construction of the hybrid map. The implementation of the theoretical model presented in Chapter 3 is explained and evaluated. This chapter covers how the information of the environment is gathered, how each element of the map is built and how they behave for specific processes such as loop closure for each component of the hybrid map.

The construction of the map can be conducted in several ways. For example, a robot can be teleoperated while it builds the hybrid map. In Chapter 5, an exploration method that allows to autonomously build the hybrid map is presented. Related work in autonomous exploration is reviewed and the proposed method is evaluated and compared to relevant works in the field.

The initial hybrid map and the information collected by the robot will rapidly be erroneous and outdated in human environments since they are not static and prone to suffer changes. For this reason, continuous adaptation of the map is a requirement for a robust long-term operation. Chapter 6 includes the proposed method for map adaptation in non-static environments. Firstly, a revision of the related work in map adaptation is presented and, to finish the chapter, the proposed method is evaluated.

As explained in this introduction, a map of the environment is just a tool so the robot can perform other tasks in an environment. In Chapter 7, we show how the proposed hybrid map can be used for topological localization, metric localization and path planning. These tasks will benefit from specific characteristics of the hybrid map.

Finally, Chapter 8 concludes this thesis giving the reader a general vision of the achievements and contributions, as well as, the open lines that will be tackled in the future.

1.5 Evolution of this Thesis

The document of this thesis is arranged following the coherence of the topic focusing, first, on hybrid maps and their construction, then, on maintaining the maps in the long-term in order to finish with some applications of the presented hybrid map. This arrangement differs from the chronological order of the developments during the years of this thesis. For this reason, I would like to take this space to explain how the thesis evolved during the years until it resulted in the present document. A graphical representation relating the document and the evolution of this thesis is shown in Figure 1.1.

This PhD thesis started as a continuation of the developments of my master thesis titled *Topological Navigation System applied to a Developed Robotic Platform* that consisted mainly in the development of a topological mapping system to perform graph-based navigation. In that work, we were dealing with mapping and path planning so the next natural step was to implement a localization method that could operate with the given topological representation of the environment. This first development is included in Chapter 7.1. At that state of the work, the maps were built while the robot was teleoperated, so the mapping process was not autonomous. Thus, our next step was to make the mapping process autonomous through an exploration algorithm, that it is presented in Chapter 5 in this document.

Our two aforementioned works inspired us to develop a hybrid representation of the environment. Initially, we saw the potential of including a higher-level representation that consisted of rooms given that they would be easy to extract with our autonomous exploration algorithm and very useful for the topological localization. So, at this point, we started conceptualizing the hybrid representation that is presented in Chapters 3 and 4. However, the conception of the hybrid map in the way that it is presented in this document has been dynamic, including new elements with the subsequent works. In order to validate the first approach to a hybrid map, we improved the topological localization method combining information from different components of the map. This is presented in Chapter 7.1.

During the first research visit and collaboration with an international lab (ASL in ETH Zurich), the metric representations were added to the hybrid map. In addition, we developed a hybrid path planning method that takes advantage of the topological and metric representations in order to perform more efficient path planning. This development is presented in Chapter 7.3. The construction of the hybrid map through autonomous exploration and the hybrid path planning method led to a publication in the International Conference on Robotics and Automation (ICRA).

At that stage, the map representation was meant for static environments and we become interested in how to enable mapping in non-static environments. During

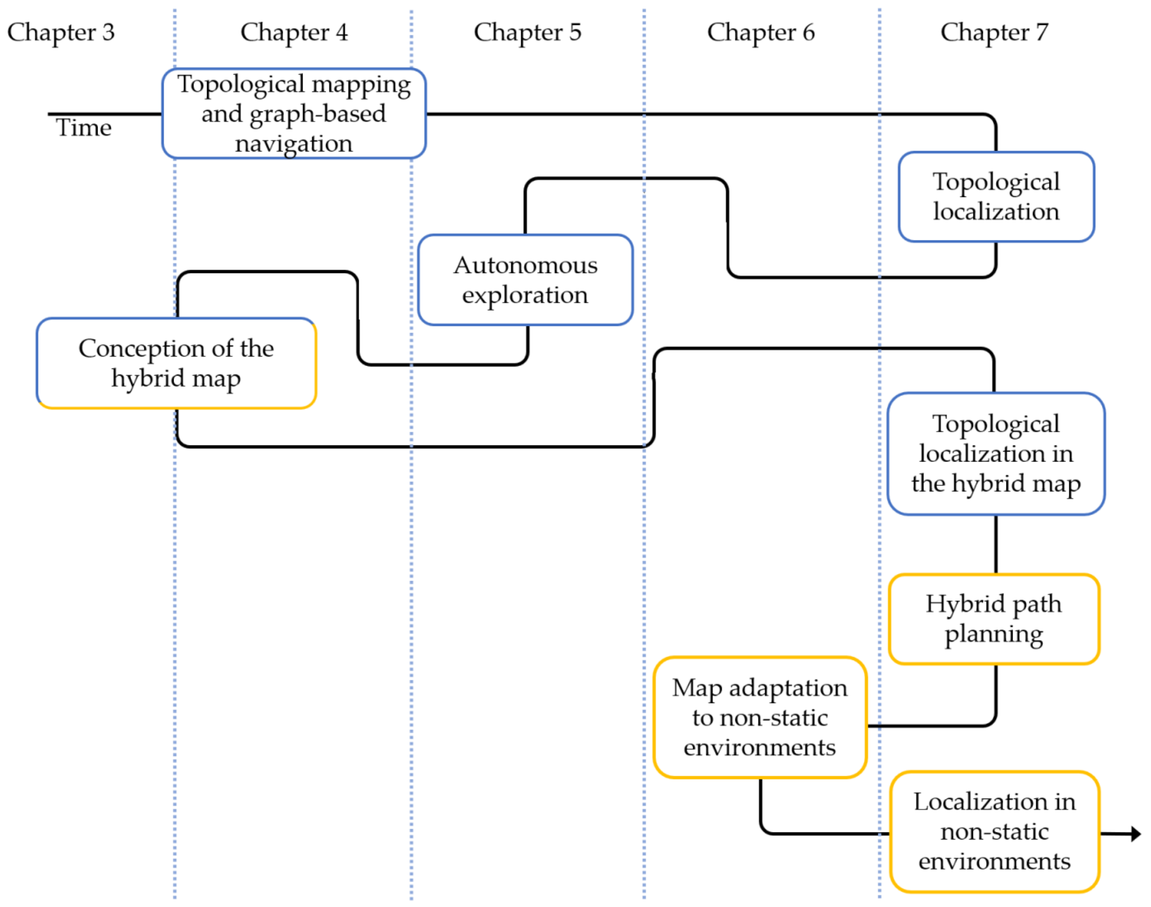

Chapter 3
Chapter 4
Chapter 5
Chapter 6
Chapter 7
Time
Topological mapping and graph-based navigation
Topological localization
Autonomous exploration
Conception of the hybrid map
Topological localization in the hybrid map
Hybrid path planning
Map adaptation to non-static environments
Localization in non-static environments

Figure 1.1 Graphical representation of the evolution of this thesis. From left to right, the chapters of this thesis are listed. The black line connects the different topics in the chronological order that they were developed. Yellow boxes indicate the topics that are a result of a collaboration with other research labs.

the second research visit (CIIRC in CTU Prague), we noticed that objects were key elements representing the dynamics of an indoor environment. Thus, we included an object-based representation in the hybrid map and perform map adaptation to changes in the environment. This development is presented in Chapter 6 and yielded a publication in the International Conference on Intelligent Robots and Systems (IROS) and in Robotics and Automation Letters (RA-L).

The last work developed for this thesis was also conducted in an international collaboration (Photogrammetry and Robotics Lab in University of Bonn). The aim of this work is to develop a robust localization method for non-static indoor environment using metric and semantic information to overcome the changes in the environment. This work is included in Chapter 7.2 and has not been published yet.

As we have described and shown in Figure 1.1, the document structure highly differs from the evolution of the thesis. However, we believe that the chosen structure will be more coherent and easy to follow by the reader.

2

Related Work in Hybrid Maps

Hybrid mapping of the environment for robot navigation has been extensively researched in the last decades. Hybrid maps were firstly proposed as a way to strengthen map representations and compensate the drawbacks of both metric and topological approaches in representing the environments [151]. In that work, Thrun stated that grid-based (metric) approaches are inefficient for planning, space-consuming, they require accurate determination of robot pose and offer a poor interface for most symbolic problem solvers. By contrast, he stated that topological approaches are difficult to construct and maintain in large-scale environments if sensor information is ambiguous, they may yield suboptimal paths when path planning and recognition of places may often be difficult. The combination of both paradigms offer the best of both worlds since one representation compensates the drawbacks of the other.

Nowadays, the drawbacks that Thurn identified in [151] are still a problem and in some cases they have even magnified. This is the case of memory and space consumption in metric maps given that the current trend is to build detailed 3D metric maps. Although some works are focused on efficiently building 3D metric maps [80, 122], they still present limitations to handle large-scale environments. For these reasons, hybrid mapping is an up-to-date paradigm that can help us solve some of the current challenges in robot mapping and navigation.

Approximately three decades ago, authors started to be interested in the possibility of using hybrid maps for robot mapping. They developed the theoretical groundwork for current hybrid mapping, along with the first experimental evaluations and proofs of concept that validated the hybrid mapping paradigm. Kuipers and Levitt [96] proposed a four-level semantic hierarchy that organized the environment information in a sensorimotor level, a procedural level, a topological map and a metric map. They tested the model in three different navigation approaches using different input information (such as views or landmarks) and storing different information in the topological and metric maps (such as landmark covisibility or paths and places, in the topological level, and metric relations between landmarks or local metric maps, in the metric level). Based on the aforementioned work, Kuipers [93] presented the Spatial Semantic Hierarchy (SSH) that established a hierarchy for robot mapping that consisted of a control level (previously referred as sensorimotor), a causal level (previously referred as procedural), a topological level and a metric level. The control level acquired 2D local maps of the surroundings of the robot, the causal level identified the views and actions performed by the robot and built a view-graph while the robot moved, the topological level contained the places, grouping views and paths traveled by the robot, and the metric level contained the 2D global map built from the local metric maps. Similarly, Duckett et al. [52, 53] proposed to acquire 2D local grip maps and a topological map to, afterwards, build a consistent global metric map. Bulata and Devy [30] presented a landmark-based hybrid map that consisted of a geometrical level, a symbolic level and a topological level. The geometrical level was formed by the landmarks extracted from sensor information. The symbolic level contained areas that were defined as semantical entities (rooms, corridors,...) that contained landmarks. Finally, the topological level represented the whole environment and contained the relationships between areas in a global frame of reference. This work established the connection between areas when crossing doors. In the same spirit, in [176] the environment is modeled as a topological map connecting places (rooms) and a 2D occupancy grid is built for each place. Different places were identified by gateway detection and gateways were detected like gaps in sensor readings of a predetermined size. Thrun et al. [151, 152, 154] presented a hierarchical mapping method consisting of a global topological and a global metric map. First, they built a 2D occupancy grid of the environment from which they, then, extracted a topological map using Voronoi diagram method. Thanks to the identification of critical lines in the Voronoi diagram (areas that minimize the clearance locally), they partitioned the global metric map and extracted the topological map of the environment. This partitioning method divided the environment according to doorways and narrow areas within rooms. Both, the metric map and the topological map, were used for path planning. The results from some of these pioneering developments in hybrid mapping are included in Figure 2.1.

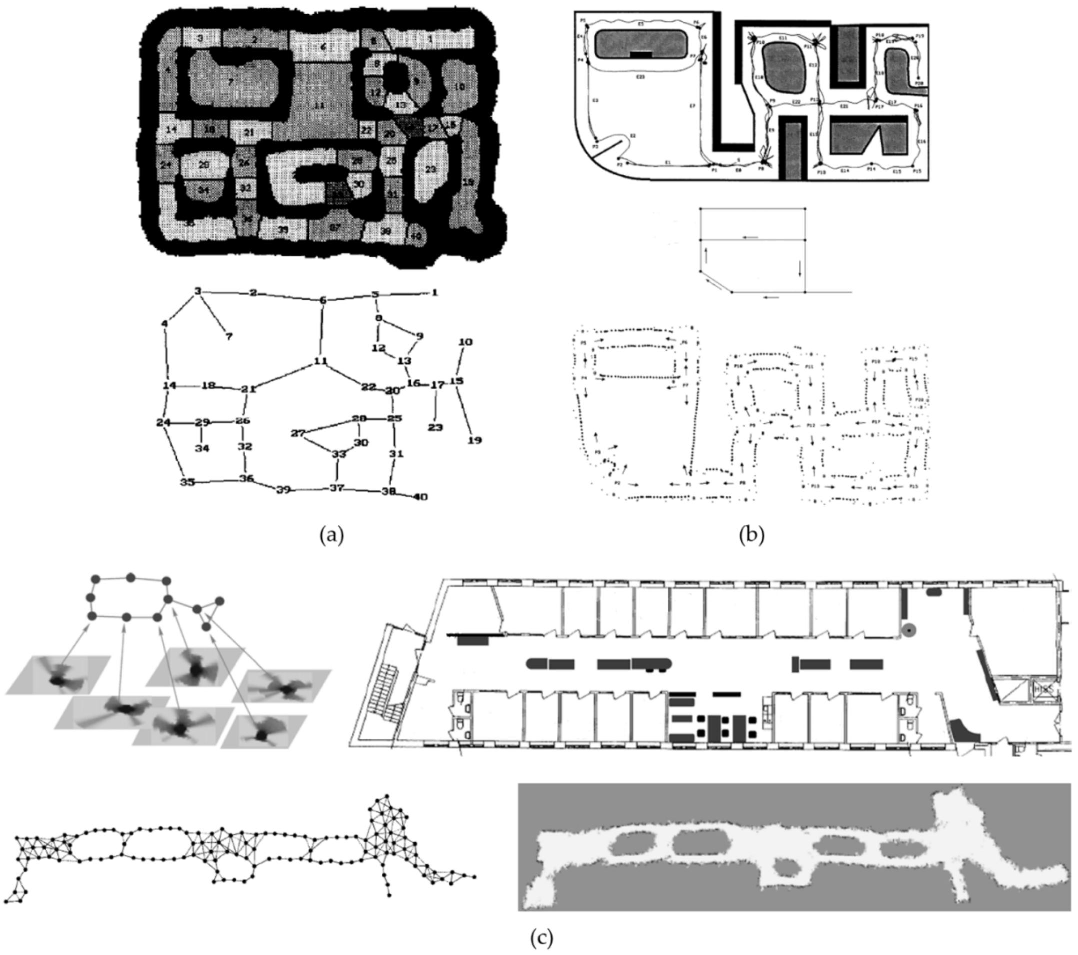


Figure 2.1 Hybrid maps obtained with the pioneering hybrid mapping methods. In (a), the map built by Thrun [151] is shown. The hybrid map consists of a 2D occupancy grid of the environment and a topological map. The global occupancy grid is partitioned according to the topological map splitting the environment where narrow passages are detected. The hybrid map built in SSH by Kuipers [93] for the path traveled in the given environment is shown in (b). A section of the topological map is included where dots represent places and lines paths connecting them. The global metric map with the landmarks detected is shown. In (c), the map built by Duckett et al. [52] is shown. A section of the topological map with the 2D local occupancy grids associated to each node is included. In addition, the global topological map and the merged global metric map are shown.

Later, some authors took over the ideas of those first hybrid mapping methods to perform more complex mapping strategies or to apply hybrid maps for higher-level tasks. An example of these approaches is Atlas framework [27, 28] proposed by Bosse et al. that consisted in a SLAM framework based on hybrid metrical/topological maps. The representation consisted in a graph of multiple local maps that could later be merged to obtain a global metric map. The local metric maps were bounded by the complexity of the map by defining a maximum number of features to be contained in each map. They proposed a path planning and a loop closing method within the Atlas framework. In [61, 66], authors proposed the Multi-AH-graph, a multihierarchical representation of large-scale spaces. In [66], they presented a two-hierarchy multihierarchical representation that consisted of a spatial representation and a conceptual one. The spatial hierarchy contained a topological map that identified rooms and corridors connected by the possibility to navigate among them and grid maps associated to each node (extracted from a global metric map). The conceptual hierarchy contained four different levels that determined the level of abstraction of the semantic information. Things were identified in the first level, then, they were categorized into objects and rooms, the third level included the specific objects and rooms and the last one the instances of objects and rooms. In [61], Multi-AH-graph is augmented to drive a wheelchair by including a localization hierarchy and several route-planning hierarchies to assist the navigation. In addition, some authors based their works on some of the pioneering hybrid mapping strategies. This is the case of [95, 117] that proposed some extension works of the SSH. Kuipers et al. [95] extended the SSH to resolve global structural ambiguity. To do so, they built local maps of perceptually complex areas such as gateways and intersections and maintained a topological map containing the different local maps to robustify the metric representation of the environment. In addition, the work in [117] studied the uncertainty of the previously mentioned extension of SSH in local metric uncertainty, global topological uncertainty and global metric uncertainty in order to build more accurate representations. Buschka and Saffiotti [32] detected an increasing interest in hybrid mapping and a lack of a systematic analysis of the different hybrid mapping methods proposed up to then, given that some works were focusing on hierarchical structures [93], others on flat ones [52], some were building the topological map on top of metric maps [154] and others were doing the reverse [53]. In order to fill that gap, they proposed a formal definition of a hybrid map and a classification of hybrid maps (we will talk about their definition and classification in Chapter 3).

Different approaches were then proposed towards hybrid mapping. The next works are going to be classified according to the structures created for mapping. Firstly, we present works that built both a global metric map and a global topological map [16, 67, 83, 179, 180, 186]. In some works, such as the work by Zhang [179], a global metric map is built and afterwards a topological map is extracted from the metric structure. In that case, Voronoi diagram was used to obtain the topology of the environment. This approach maintains the global metric map and the global topological map and search-based planning in the hybrid map was presented in their other work [180]. On the contrary, in [16], Beeson et al. built the map representation reversely, in the first place they built the topological map and they constructed the global metric map on top of it. The global metric map is built upon the skeleton of the topological map and they also built metric local maps for gateways and intersections. In [186], Zivkovic et al. studied the creation of a geometric global map (2D occupancy grid) or an appearance-based map that consisted of a collection of sensor readings from which authors extract a topological map from any of them. Then, they performed normalized graph cut to divide the graph in node clusters, each node cluster was associated to a node of a higher-level graph in order to perform enhanced path planning strategies. Normalized graph cut consists in dividing the graph through weakly connected places, that might correspond to doorways or narrow areas. In [83], SLAM was used to build a global topological and a global metric map. A path planning strategy on top of the topological map was used to autonomously build the maps of the environment through exploration while the metric map was used for localization.

The next works built a topological map from a global metric map and then partitioned the metric map into local metric maps [22, 62, 84, 129, 138]. Local metric maps offered a more tractable detailed representation of the environment that leaded to more efficient map building and path planning. An extension of the previously-described work [83] was proposed in [84]. In the previous work, an occupancy grid of the environment and a topological map had been built. In this extension, a Voronoi diagram was extracted from the occupancy grid to partition de environment through a normalized cut method in which the topological map was divided according to narrow passages. Afterwards, local metric maps were extracted from the global map for each area. In [138], Schmuck et al. mapped the environment as a global metric map using Octomap algorithm [80] and extracted its topology. They divided the global metric map into 3D fixed-size submaps in order to enhance map optimization and path planning. Pronobis et al. [129] proposed a semantic mapping system that comprised four layers: a sensory layer that contained the metric map of the environment, a place layer that organized the environment topologically into places and paths, a categorical level that included objects and landmarks and a conceptual layer that contained relations between elements. They built a global metric map and a global topological map and grouped different topological places into rooms by detecting doorways. Although these works generate submaps, their main problem is the computational requirements to build the global metric map, specially in large environments. The solution to this problem leads to the third group of hybrid mapping approaches.

The following works, that belong to the third group, directly built the global topological map and the local metric maps without the need of a global metric map [21, 46, 57, 63, 90, 100, 120, 127, 128, 156, 159, 173]. Estrada et al. presented Hierarchical SLAM in [57]. Their hierarchical mapping method consisted of a global topological map and feature-based local maps of limited size. Similarly to other methods, each node of the topological map was associated with a submap and the creation and size of the submaps depended on the complexity of the submap (maximum number of features) and on the uncertainty in robot localization. In [159], each node of a topological map contained the set of images in that region and it was augmented with metric information at places (nodes) where the robot turned. Similarly, in [156, 46, 173], metric information was not stored for the whole environment. In [156], Tomatis et al. presented a localization and map building method based on a topological global map and they added local metric maps only for some specific areas of the environment. In [46], Dayoub et al. presented a method in which each of the nodes of a pose graph was associated to a 3D point cloud and the point clouds of interesting areas such as intersections could be merged together in a 3D occupancy grid. In [173], metric maps were only created for complex areas, whereas corridor and simple areas were maintained through a topological map. Nitsche et al. [120] proposed a hybrid mapping method consisting in a topological map and fixed-size metric submaps. Each metric map, built as an occupancy grid, was associated to an area node of the topological map. The topological map also contained gateway nodes that represented the section of an occupancy grid edge where two local maps were connected. The method proposed by Blanco et al. [21] simultaneously built the topological map and the local metric submaps. Submaps were divided according to the covisibility of observations forming areas (nodes) that contained observations that would be likely observed together. In the same spirit, Forster et al. [63] generated submaps according to the detection of interconnected RFID tags. A metric and a topological map were built for each cluster of RFID tags and they were also associated to a node of a higher-level topological graph. Konolige et al. [90] presented a topological graph overlaid with fixed-size local occupancy grids. They maintained two topological maps that were the pose graph and the navigation graph. The pose graph associated robot poses with sensor data, while the navigation graph contained the traversable edges between nodes. Figure 2.2 shows an example of the resulting hybrid map for one representative work of each of the mentioned groups (approaches building global metric and topological map, approaches building global topological map and global and local metric maps and approaches building global topological map and local metric maps).

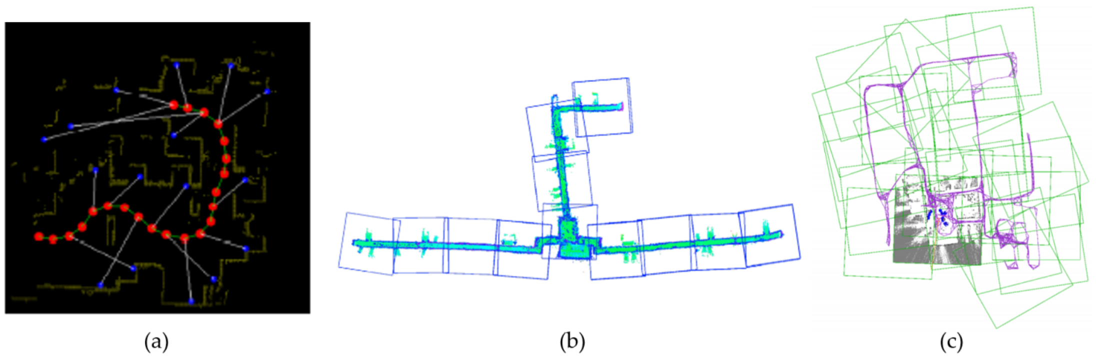
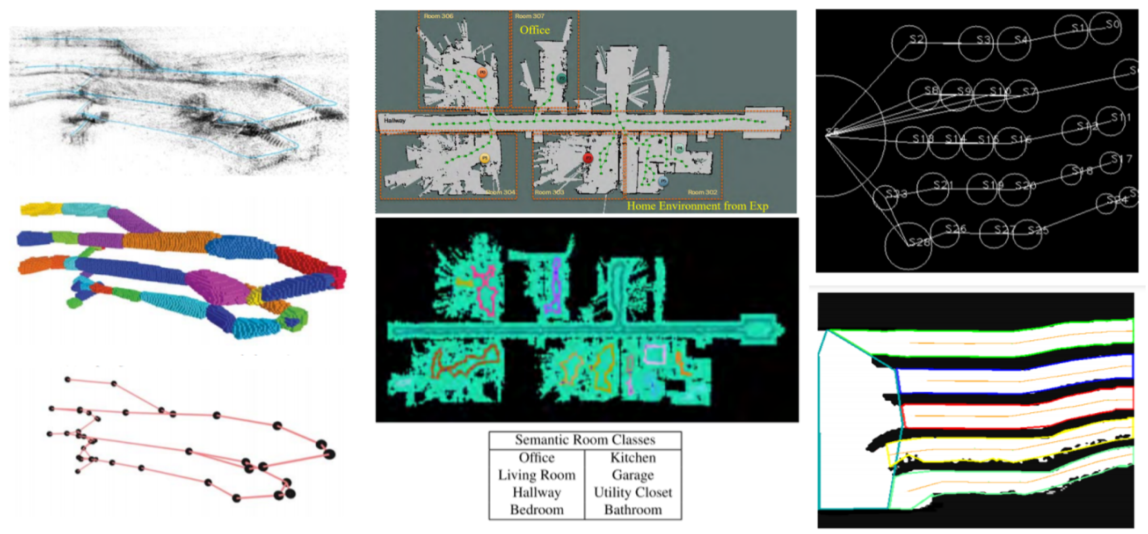


Figure 2.2 Hybrid maps obtained for some relevant works. In (a), the resulting map built by Jia et al. [83] is shown. This representation consists of a global metric map and a global topological map. The work by Schmuck et al. [138] is shown in (b) and the environment is represented as a global topological map and a global metric map that is then partitioned into submaps according to topological nodes. In (c), we show the resulting map built by Konolige et al. [90] where metric submaps and a global topological map are simultaneously built.

Hybrid maps are also receiving attention from the robotics community recently and some relevant works have been proposed during the last years [13, 23, 38, 78, 109, 135, 136, 149, 150]. Blochliger et al. [23] present a topological representation that maps the free space into a navigation graph and 3D convex voxel clusters. A node of the navigation graph is assigned for each convex area and the adjacency between areas is mapped through portals in the navigation graph. Convex areas are identified as they offer safe areas for path planning. In [109], Luo et al. present a hybrid topological-metric map with semantic information. Semantic information in the object level generated from a CNN is stored in the topological nodes of the hybrid map. Room types in the environment are inferred from the information of objects contained in the nodes. Both, the 2D global metric map and the topological map, are segmented into areas and clusters of topological nodes based on the room type. The aim of [78] is to build a globally deformable map for underwater navigation. For this purpose, they propose a hybrid map that consists of 3D metric submaps connected through a pose graph. The size of the submaps is computed by keeping the pose covariance of the robot below a maximum value. In [135, 136], Santos et al. propose a hybrid mapping for robot navigation in an agricultural environment. They use satellite images to build a 2D global occupancy grid of the environment and extract its structure using Voronoi diagram. They build a topological map of the traversable space in a vineyard and partition the global metric map into occupancy grids for common areas and vineyard rows. For efficient path planning, the algorithm focuses on the interesting areas by only selecting the occupancy grids of the regions of interest. Figure 2.3 shows the resulting hybrid maps for some of the aforementioned recent works in hybrid mapping.



Semantic Room Classes	
Office	Kitchen
Living Room	Garage
Hallway	Utility Closet
Bedroom	Bathroom

(a) (b) (c)

Figure 2.3 Some relevant recent works in hybrid mapping. In (a), the resulting map for the work proposed in [23] is shown. They build a 3D voxel-based map for each convex area in the environment and a navigation graph that connects the adjacent convex areas. The raw information of the environment captured as a point cloud is shown in the top figure. In (b), we show the resulting hybrid map for the work by Luo et at. [109]. The first figure shows the 2D occupancy grid of the environment, the topological map and the partition of the environment into rooms. In the next figure and the table, the room types identified are included (different colors for each room type). In (c), the resulting hybrid map for the work by Santos et al. [135] is included. They build a topological graph of the traversable areas in the environment and, then, divide the 2D occupancy grid into regions of interest determined by the rows and common areas of the vineyard.

The presented hybrid mapping methods mainly construct topological and metric representations of the environment. Most of them keep a global topological representation and represent the environment locally into metric submaps, although some opt to maintain both representations globally [16, 83]. The main drawback of building a global metric representation is the high memory consumption of the map, that will also propagate to long processing times when using the map. The approaches that build local maps of the environment mainly differ in the construction of the topological map (using Voronoi diagram [84, 136, 154], as a pose graph [46, 78, 90, 149], as a navigation graph [23, 90, 109] or merely as anchoring for the submaps [66, 38]) and the metric submaps (as 2D occupancy grids [53, 90, 109], as 3D voxel structures [23, 78, 138], as landmark maps [30, 63, 93] or raw sensor information [46, 149, 150]). In addition, different approaches for partitioning the environment have been presented (covisibility of elements [21, 63], complexity of the submap [28, 57], uncertainty in pose covariance [57], fixed-size submaps [90, 120, 138] or environment-related submaps such as convex areas [23], rooms [109, 129, 176] or doorways and narrow passages [95, 186]). Different partitioning strategies have different advantages and the decision will be highly related to the task to perform. However, partitioning the environment according to rooms is the representation that offers a higher semantic understanding of the environment and eases human robot interaction. Furthermore, some approaches present complex hierarchies with several levels [30, 61, 96, 129] while other are based on a flat structure in which a single topological map acts like a global map for the multiple generated submaps [13, 27, 78, 120, 151].

The methods that are more alike to the one proposed in this thesis are [23, 109, 176]. Similarly to Blochliger et al. [23], we propose the construction of 3D voxel-based maps that are connected through a topological map. In addition, we also build a representation similar to their navigation graph that contains information of traversable areas and portals in the environment. In a similar spirit as Luo et al. [109] and Youngblood et al. [176], we divide the environment into rooms and build a metric submap for each one. In addition, we assign a node of the topological map to each room. We are also related to Youngblood et al. [176] in the way of identifying rooms using gateways and to Luo et al. [109] since we also relate object and room type information to topological nodes. Differently from these works, we present a four-level hybrid map structure that distributes all the mentioned information in different levels. To the best of authors knowledge, our hybrid map is the only one to contain 3D submaps, navigable paths information and objects and scenes information distributed into different components. As we will describe in the detail in the following chapters of this thesis, our hybrid representation consists of three topological maps and multiple metric submaps. The topological graph contains a node for each room of the environment associated with its room type and the edges of the map refer to the doors connecting the rooms. The traversability graph contains information of the free areas in the environment and acts as a navigation graph that also contains door information. The object-based pose graph adds semantic information by relating the traversability graph with the objects in the environment. Multiple 3D metric maps are also built, one per room in the environment, that will allow safe path planning.

Once we have introduced the most relevant related works in the field of hybrid mapping for mobile robots, we are in a good starting position to describe the hybrid mapping method proposed in this thesis and understand its novelties and advantages compared to the current state of the art of the field.

3 Model of the Environment

Models of the environment are needed for a wide range of robotic applications and they should serve for defining in the greatest detail an environment so a robot is certain about the actions to take in order to reach a goal or perform a task. For example, the map of a kitchen should contain information so that the robot can safely move in it, it should provide means to know where the different cooking elements are or how to look for them. In addition, it should give the information of how the kitchen is connected to the rest of the house in case it has to move elsewhere. We could also think that the map should contain information of how the kitchen behaves in the long term and which elements change more than others. A map could also contain information about how to interact with objects, how users interact with objects and the list of actions to perform in each area of the kitchen. As shown, there are many tasks for which the map could be helpful. In this thesis, we want to focus on providing the robot with a map that allows it to move safely in the environment, to identify the general structure and the distinguishable elements of the environment. Moreover, we want to give the robot the capability to build a detailed reconstruction that, in the future, will allow the robot to perform higher-level tasks. In addition, to fulfill the goal of autonomous robots, we want our map to allow the robot to detect anything that has changed over time and correct it, so it does not have to remap the environment every time something changes.

Currently, there is not a map representation that allows us to perform all the desired tasks. Metric approaches such as occupancy grids allow the robot to compute optimal paths and to localize accurately. However, they are hard to create and maintain due to inaccuracies, they involve heavy computation costs and do not interface well with human symbolic thinking. By contrast, topological maps scale well and interface more naturally with humans. However, they offer more limited localization and path planning capabilities. With the above description of metric and topological approaches, we see that we could use a metric approach to obtain a detail representation of the environment, map the different elements of the environment and move safely in it. Due to the computational costs, finding feasible paths can become complicated in large environments in real time. In addition, they do not offer an intuitive representation of the structure of the environment. Regarding topological maps, we could use them to easily represent the structure and the elements of the environment and compute rough safe paths. However, they do not provide detailed information of the environment not even for path planning. As shown, different representations are more suitable for different tasks. For this reason, hybrid maps, that contain metric and topological representations, have been considered as a promising solution in the last decades. Hybrid maps give the chance to exploit the advantages of both representations, combining the accuracy and detailed information of metric maps and the robustness and scalability of topological maps.

In this chapter, we present a novel structure for hybrid maps, that is the core representation on which this thesis focuses. We will see the importance of representing and organizing carefully the information in order to enhance the functionality of the model.

3.1 General Overview of the Hybrid Map

Prior to defining the hybrid map structure proposed in this thesis, we need to introduce several concepts and common terminology. Firstly, the concept of map for a robotic system needs to be addressed in order to, afterwards, define what a hybrid map is and which is our specific proposal for the structure of this map.

Several authors have previously defined the concept of map for a robotic system. In [32], Buschka and Saffiotti define a map as *any digital representation of the space*. Previously, in [154], Thrun et al. defined a map of the environment as *an assignment of properties to each x-y location in the world*. In [153], Thrun, Burgard and Fox defined a map of the environment as *a list of objects in the environment and their locations*.

In addition, in [82], Chatila defined a map as *a representation of the environment, or parts of it*. Although being different definitions, if we stick to their similarities, a map is simply a representation of the environment. In this way, all the instances of representations of an environment that one can think about, are included in the concept of map.

Given the definition of map, we can intuitively expect a hybrid map to be a representation of the environment that combines or mixes other representations or information from several representations. In [32], Buschka and Saffiotti also give a formal definition of a hybrid map: *a hybrid map is a pair* $H = \langle M, C \rangle$ *where* $M = \{M_1, ..., M_n\}$ *is a set of maps, and* $C = \{c_1, ..., c_p\}$ *is a set of links. Each link is a pair* $\langle m_i, m_j \rangle$*, where* m_i *is an element of* M_i *and* m_j *is an element of* M_j*, with* $i \neq j$. Each of the maps that constitute the set of maps of the hybrid map will be referred as component. A hybrid map is supposed to contain different information in each of its components whose correspondences are determined by links of the hybrid map. For example, one of the components can contain information about laser measurements and the other component information about objects. In this case, the links between these two components would identify which laser scans correspond to which object. With this simple example, we can observe that one of the advantages of hybrid maps is the connectivity or correspondence between components. Without this connectivity, a hybrid map would simply be a collection of several independent maps.

A taxonomy of hybrid maps along three different dimensions is also presented in [32]. They state that a hybrid map can be classified according to its heterogeneity, hierarchy and separability as follows:

- Heterogeneity refers to the type of map of the components of the hybrid map; if components are of different types or contain different type of information the hybrid map is heterogeneous, while if all the components are from the same type it is homogeneous.
- Hierarchy refers to the ordering between components, being hierarchical if one component represents the environment at a higher abstraction level than other.
- Finally, separability refers to the independence of each component. If all the components are self-contained and independent the hybrid map is separable. On the contrary it will be integrated.

They concluded that heterogeneity is a requirement for a hybrid map, while hierarchy and separability are desirable. At the end of this chapter, the proposed hybrid map will be discussed according to this classification and the possible synergies between components.

According to the definition of hybrid map given previously, the hybrid map presented in this thesis can be defined as $H = \langle\{T_1, T_2, T_3, M_1, ..., M_n\}, C\rangle$ where T_1, T_2 and T_3 are topological maps and $M_1, ..., M_n$ are geometric maps.

Three topological maps are included as components of the hybrid map since they contain different interconnected information. For the seek of clarity, let's define T_1 as the topological map TM of the environment that contains the information about the different places of the environment and their connections; T_2 as the traversability graph TG that represents the traversable paths in the environment; and T_3 as the object-based pose graph OP that represents the objects seen in the environment and the poses from where those objects were seen. We decided to keep these representations as different components of the hybrid map because they offer different possibilities for localization and path-planning and they also behave differently towards changes in the environment (as will be described in Chapter 6).

The number of geometric maps in our hybrid map will depend on the environment since one geometric map is built for each place detected in the environment. 3D information is included in the geometric maps in order to have detailed information of each place.

Taking into account the above clarifications, we can define our hybrid map as $H = \langle\{TG, TM, OP, M_1, ..., M_n\}, C\rangle$. A conceptual representation of our hybrid map is shown in Figure 3.1 where each of the components for a given environment is included. Let this figure be an illustrative example of the different components that will be explained along this chapter. In that sense, in the rest of this chapter, the different elements and characteristics of the components of the hybrid map are described in detail and also the connections between components.

3.2 Topological Map

The topological map, $TM = \langle S, T\rangle$, refers to the topological representation of the structure of the environment and it is the higher abstraction level component of the proposed hybrid map. The topological map contains information about the different places that integrate the environment and the connections between them. It gives a general representation of the structure of an environment whose detailed representation will be provided by lower level components. It is a toposemantic component in which nodes are called supernodes, S, that are connected between them through transitions, T.

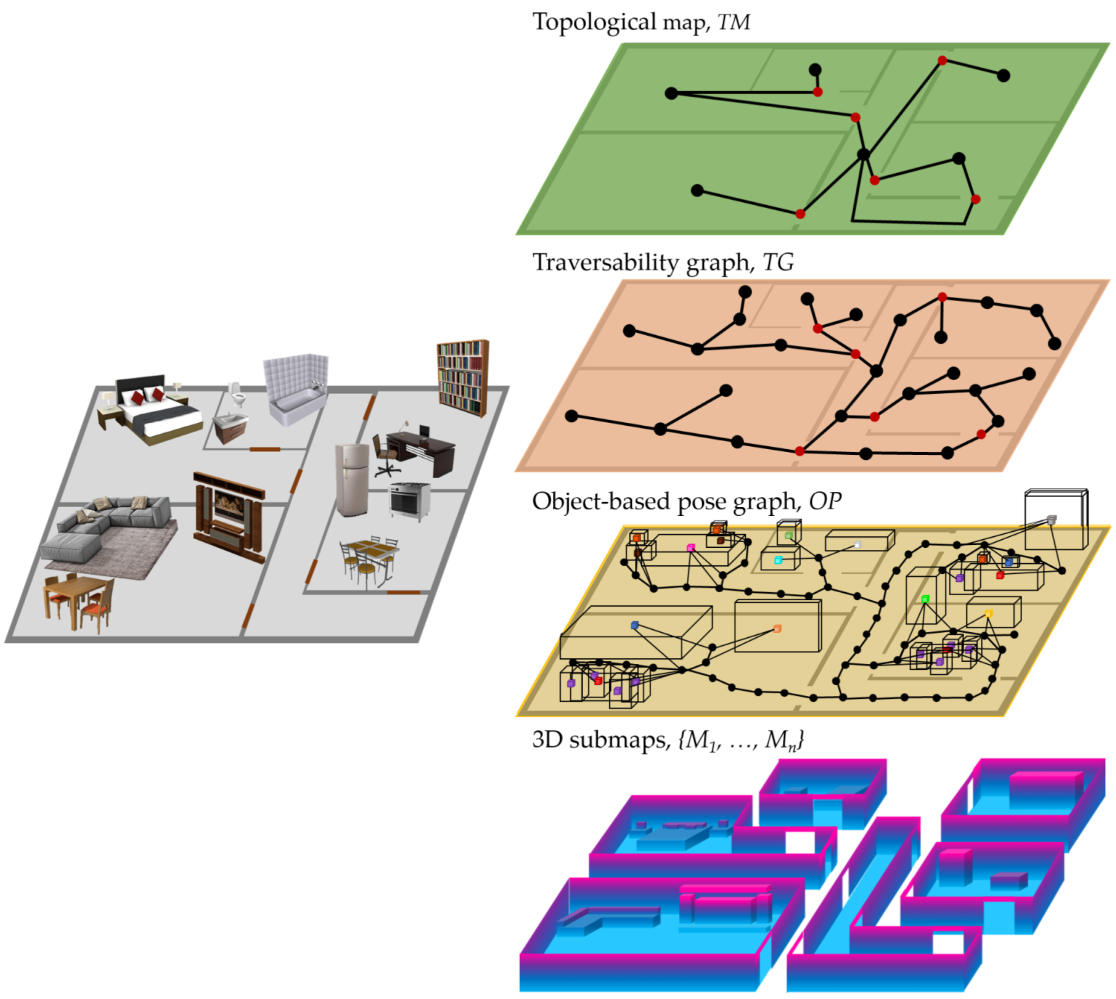

Topological map, *TM*
Traversability graph, *TG*
Object-based pose graph, *OP*
3D submaps, $\{M_1, \ldots, M_n\}$

Figure 3.1 Example of indoor environment and the hybrid map that will result from mapping it. Each of the components of the hybrid map are shown: the topological map, the traversability graph, the object-based pose graph and the six 3D submaps generated for each room.

Each supernode, s_i corresponds to a distinctive place in the environment. As the scope of this thesis are indoor environments, we can assume that distinctive places are rooms. Therefore, each transition, $t_{i,j}^{[k]}$ that connects two different supernodes i and j is a door whose identifier is k. In this sense, in an indoor environment the topological map contains the information about rooms and their connections through doors.

Each supernode and transition of the topological map contains relevant information about the structure of the environment and its semantic interpretation. Supernodes are firstly identified with a unique key, s_i^{id}, that is also related with the number of rooms (N_s) in the environment, $N_s = \max(s_i^{id})$. Each supernode is also characterized with its semantic class, $class_i$. The semantic class of a supernode refers to the scene type[1] that corresponds to that room (i.e. kitchen or living room). As supernodes are high-abstraction representations, they also contain references to the elements of the lower abstraction components that belong to that supernode. This includes references to the identifiers of nodes of the traversability graph, $\{v_i^{id}\}$, and poses, $\{w_i^{id}\}$, and objects, $\{o_i^{id}\}$, of the object-based pose graph. The topological map just contains a reference to the key of the corresponding nodes, poses and objects, further information about them is included in Sections 3.3 and 3.4. Finally, each supernode is associated with the supernodes that it is connected to (also referred as priors). A supernode connected to another given supernode s_i is referred as $\dot{s}_i$. This information is included as a vector of keys, $\dot{s}_i^{id} = \{\dot{s}_i^1, ..., \dot{s}_i^n\}$, since normally one supernode is connected to more than one other supernode. Then, each supernode can be identified by its parameters as $s_i = (s_i^{id}, class_i, \{v_i^{id}\}, \{w_i^{id}\}, \{o_i^{id}\}, \{\dot{s}_i^{id}\})$.

Transitions represent the connection between two supernodes. As said before, supernodes are associated with rooms in the environment, therefore, transitions are associated with the doors that connect rooms. Each transition, $t_{i,j}^{[k]}$, is uniquely identified with a key, t_k^{id}, and contains the reference to the supernodes s_i and s_j that it connects. In addition, since they represent doors and every door is also mapped in the traversability graph (as it is explain later), transitions also contain information about the node of the traversability graph assigned to the door, $d_k^{[l]}$, where l refers to the node index and k to the transition index (different indexes are needed for transition and node given that the node of the traversability graph 14 can correspond to the door 3). The information contained in each transition can be summarized as $t_{i,j}^{[k]} = (t_k^{id}, s_i^{id}, s_j^{id}, d_k^{[l]})$. An example of a topological map and different supernode and transition definitions is shown in Figure 3.2. The information contained in different supernodes and transitions is shown along the graph. For example, $t_{1,5}^5$ refers to the 5^{th} door detected in the environment, that connects supernode 1 (the hallway) with supernode 5 (the office) and whose node key is 26. An example of supernode is s_3, whose key is 3 and semantic type 1 (bedroom). That supernode contains 5 nodes (20, 21, 22, 23 and 24), 5 objects (17, 18, 19, 20 and 21) and 13 poses (34, 35, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47 and 48) and is connected to supernodes 1 (the hallway) and 4(the bathroom). Although a graphical representation of the graph is shown in Figure 3.2, the topological map is stored as a text file including the list of supernodes and transitions and their parameters.

[1]The semantic class is obtained through a scene recognition method [77] that has been developed in other thesis of this group and is beyond the scope of this thesis.

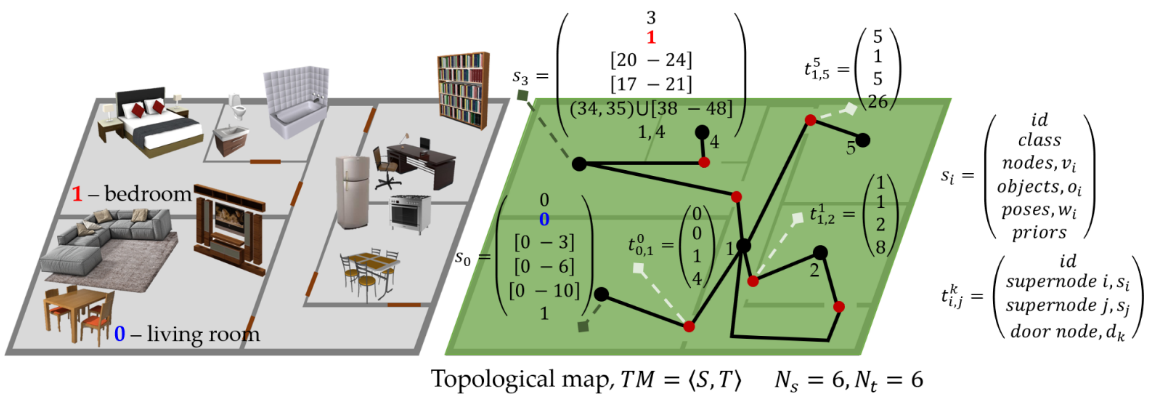

1 – bedroom
0 – living room
$s_3 = (3, 1, [20-24], [17-21], (34,35)\cup[38-48], 1,4)$
$t^5_{1,5} = (5, 1, 5, 26)$
$s_i = (id, class, nodes, v_i, objects, o_i, poses, w_i, priors)$
$s_0 = (0, 0, [0-3], [0-6], [0-10], 1)$
$t^0_{0,1} = (0, 0, 1, 4)$
$t^1_{1,2} = (1, 1, 2, 8)$
$t^k_{i,j} = (id, supernode\ i, s_i, supernode\ j, s_j, door\ node, d_k)$
1
2
4
5
Topological map, $TM = \langle S,T \rangle$ $N_s = 6, N_t = 6$

Figure 3.2 Topological map of an illustrative example of indoor environment. Black dots in the representation correspond to supernodes of the topological map and red dots represent the transitions. For this environment, six supernodes (N_s) and transitions (N_t) are created. Information about different supernodes and transitions is given according to the templates in the right. Supernode information includes the supernode unique key, its class (for this example, the supernodes corresponding to living room and bedroom are shown), the nodes, poses and objects that belong to that supernode (this information is shown in detail in Figures 3.3 and 3.4) and the supernodes that this supernode is connected to. Transition information includes the transition identifier, the two supernodes that this transition connects and the door node that relates to this transition (this information corresponds to the doors shown in Figure 3.3).

The topological map represents the invariant structure of the environment according to its rooms. We refer to it as invariant structure because in most indoor environments the structure is likely to remain and only under exceptional situations this structure will change. Closing doors is not interpreted as a change in the structure of the environment as, although the robot cannot visit that part of the environment in that moment, in the future that area will be available again. This situation will be managed through other components of the hybrid map. For this reason, this component is assumed to be static and stable to changes in the environment.

There are several applications in which the topological map presents an advantage for the system, namely for localization and human robot interaction. The topological map contains scene information for each room that helps in localization disambiguation. We can imagine a situation in which the robot has several guesses of its location given its sensorial information. In such a situation, scene information could rise the probabilities of the guesses that match with the scene type obtaining a better estimate of the robot position (this is further illustrated and discussed in Chapter 7). In addition, this information is also useful in the interaction with humans since it is information that both, the robot and the human, understand. For example, if the robot tells that it is in the kitchen, it is more valuable information than saying that it is in coordinate (7.6, 5.2)m. Moreover, it is more likely that the human tells to the robot to move to the office than giving a precise coordinate. In this sense, the higher toposemantic level of abstraction helps in localization and path-planning when semantic information is available and it becomes specially useful when humans are involved.

3.3 Traversability Graph

The traversability graph, $TG = \langle V, E \rangle$, is a topometric representation that contains the collision-free paths in the environment. In the traversability graph, a safe path is included that generalizes over the set of paths that could connect two points A and B (that are the nodes). The traversability graph does not force the robot to follow a specific path, it just indicates that between A and B there is a collision-free navigable path. The traversability graph is formed by nodes, V, and edges, E. Each node of the traversability graph, v_i, corresponds to a collision-free position that maximizes the distance to nearby obstacles. Each edge connecting two nodes, $e_{i,j}$, represents the collision-free path between two nodes.

Each node and edge of the traversability graph contains important information for the model. A node, which is uniquely identified with a key, v_i^{id}, contains its 2D pose, $\mathrm{x}_i = (x_i, y_i)$ that will allow the robot to revisit that same node. In addition, it includes the semantic class of the node, $class_i$. The semantic class defines each node as being transit node or free-area node. Transit nodes are those that connect two different places (in this case, rooms) in the environment. Along this thesis, given that it is meant for indoor environments, transit nodes are also going to be referred as doors, $D = \{d_1^i, d_2^j, ..., d_M^l\}$, where $1, 2, ...M$ refer to the door index and $i, j, ...l$ to the index of the node identified as a transit node. Free-area nodes are nodes within a same place that allow the robot to move inside the place. Each node also contains the information of the key of the prior nodes that lead to it. The prior nodes of a given node v_i are labeled as $\dot{v}_i$ and, thus, the keys of the prior nodes are $\dot{v}_i^{id} = \{\dot{v}_i^1, ..., \dot{v}_i^n\}$, given that normally one node is connected to more than one other node. Finally, each node contains the key of the supernode that it belongs to, s_i^{id}. Only free-area nodes belong to a supernode and have this parameter, since transit nodes connect two different supernodes. Regarding edges, which are also identified with a unique key, $e_{i,j}^{id}$, the information that they provide is mainly oriented to successfully moving through the graph. Firstly, an edge includes the key of the two nodes, v_i^{id} and v_j^{id}, that it connects. It also contains the behavior that has to be performed during the transition. The robot is allowed to perform several behaviors, as it will be explained in detail later, these behaviors vary mainly if the robot is moving to/from a transit node or a free-area node. Finally, each edge has a parameter to determine if the last time the robot was commanded to travel through it, it was traversable or not, $tr_{i,j}$. This will help to identify blocked edges due to changes in the environment and to find other paths to reach the navigation goals. When following a path, blocked edges are detected using a laser sensor and avoided by increasing the weight of that movement in the path planning algorithm.

To summarize, we can define every node of the traversability graph according to its parameters $v_i = (v_i^{id}, \mathrm{x}_i, class_i, \{\dot{v}_i^{id}\}, s_i^{id})$ and, similarly, every edge as $e_{i,j} = (e_{i,j}^{id}, v_i^{id}, v_j^{id}, beh_{i,j}, tr_{i,j})$. For each pair of nodes i and j, two edges are created, $e_{i,j}$ and $e_{j,i}$, as the behaviors required to reach i from j might be different that the ones required to reach j from i. Along this thesis, for simplicity we will refer to the definition of an edge, $e_{i,j}$, but we refer to both edges, $e_{i,j}$ and $e_{j,i}$. An example of a traversability graph and several examples of node and edge definitions are shown in Figure 3.3. An example of edge is given by $e_{15,25}$ whose identifier is 25 and connects nodes 15 and 25. The behavior associated with this node is 0 (moving between free-area nodes) and it is not traversable at that moment in time. This is due to an object that is blocking the path. Moreover, if the robot is at node 15 it will not be able to reach any of the nodes that belong to the office since there is not an alternative path. An example of node is given by v_{26}, whose identifier is 26 and it is located at (13.6, 7.1)m with respect to the map frame. This node is associated with a door and it is connected to nodes 25, 27, and 30. In addition, this node does not belong to any supernode, as it corresponds to a door. Although a graphical representation of the graph is shown in Figure 3.3, the traversability graph is stored as a text file including the list of nodes and edges and their parameters.

The traversability graph has to adapt to changes in the environment given that edges or nodes can be blocked by new obstacles. Assuming that the traversability graph is static would lead to failures due to collisions to unmapped obstacles. Detecting and avoiding non traversable edges is a strategy to overcome possible failures.

The traversability graph is used to plan safe paths in the environment as all the nodes and edges are collision-free areas and it is suitable for fast graph search algorithms. In addition, it is used for loop closure. The robot will detect when a node has been reached again and determine if it has been reached through a new path, so it has to

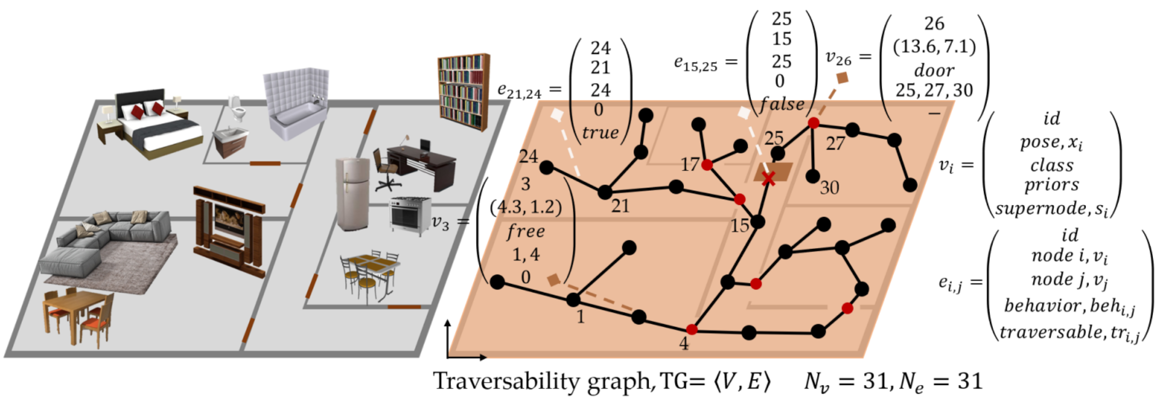

$e_{21,24} = \begin{pmatrix} 24 \\ 21 \\ 24 \\ 0 \\ true \end{pmatrix}$
$e_{15,25} = \begin{pmatrix} 25 \\ 15 \\ 25 \\ 0 \\ false \end{pmatrix}$
$v_{26} = \begin{pmatrix} 26 \\ (13.6, 7.1) \\ door \\ 25, 27, 30 \\ - \end{pmatrix}$
$v_3 = \begin{pmatrix} 3 \\ (4.3, 1.2) \\ free \\ 1, 4 \\ 0 \end{pmatrix}$
$v_i = \begin{pmatrix} id \\ pose, x_i \\ class \\ priors \\ supernode, s_i \end{pmatrix}$
$e_{i,j} = \begin{pmatrix} id \\ node\ i, v_i \\ node\ j, v_j \\ behavior, beh_{i,j} \\ traversable, tr_{i,j} \end{pmatrix}$
24
21
17
25
27
30
15
1
4
Traversability graph, TG= $\langle V, E \rangle$ $N_v = 31, N_e = 31$

Figure 3.3 Traversability graph in an illustrative example of indoor environment. Black dots in the representation correspond to free-area nodes, red dots represent the transit nodes (doors) and lines represent edges. For this environment, 31 nodes (N_v) and edges (N_e) are created. Information about different nodes and edges is given according to the templates in the right. Node information includes the node unique key, the pose of the node (x-y coordinates), its class (free for free area node and door for transit node) and the nodes connected to this node. Edge information includes the edge identifier, the two nodes that this edge connects, the behavior that the robot has to perform to move between the nodes and if the edge is detected as traversable or not.

create a new edge for this movement, or it has been reached through an already-known path that will not require any modification of the map. An example of loop closure is found in Figure 3.3 where there are two entrances from the hallway to the kitchen. The strategy for loop closure is further explained in Chapter 4.

3.4 Object-based Pose Graph

The object-based pose graph, $OP = \langle W, O \rangle$, is a topometric representation in 3D that includes object nodes and pose nodes. The object-based pose graph relates a path in the environment with its objects offering a semantically-enhanced graph. Up to now, the components included in the hybrid map only contained 2D representations regarding the structure and ground traversability. This component merges the 2D representation of the ground traversability with the 3D information of the objects present in the environment. Pose nodes (in the following referred as poses), W, track the trajectory traveled by the robot and object nodes, O, represent the 3D pose and type of the objects[2] in the environment.

Each pose contains metric information regarding the path and the poses from where objects were seen. A pose, w_i, which is identified by a unique key, w_i^{id}, contains the global coordinates of the pose, $\mathrm{x}_i = (x_i, y_i)$ and the key of the objects that were detected from it, $\{o_i^{id}\}$. Also, like most of the elements in the previous components, the poses that it is connected to are included, $\{\dot{w}_i^{id}\}$, and the supernode that the pose belongs to, s_i^{id}. Therefore, we can define each pose as $w_i = (w_i^{id}, \mathrm{x}_i, \{o_i^{id}\}, \{\dot{w}_i^{id}\}, s_i^{id})$.

One object node is added to the map each time that the robot detects a new object in the environment. Each object, o_i, is identified with a unique key, o_i^{id}. It contains information about the 3D position of its centroid, $c_i = (x_i, y_i, z_i)$ and the 3D bounding box of the object, which is identified by the width, height and length of the box, $b_{i,3D}$. Objects are also assigned an object class, $class_i$, that correspond to the most likely class of the object (chair, table, etc). Objects are also identified by their behavior in the long term. The persistence probability of an object, $p(o_i) = [0, 1]$, indicates how static that specific object has been during the visits of the robot to that environment. As we will see later, for long-term operation, we are not removing objects when they are not present in the environment. On the contrary, we lower their persistence probability and also indicate that they were not active in the last visit. For this purpose, we include a binary parameter, a_i, that indicates whether an object was present in the last visit, $a_i = 1$, or not, $a_i = 0$. Lastly, each object contains information about the poses from where that object was observed, $\{w_i^{id}\}$, and the supernode where the object is, s_i^{id}. Summarizing all these parameters, an object can be defined as $o_i = (o_i^{id}, c_i, b_{i,3D}, class_i, p(o_i), a_i, \{w_i^{id}\}, s_i^{id})$.

The object-based pose graph of an illustrative indoor environment is shown in Figure 3.4. The representation shown includes different pose and object definitions based on the templates shown at the right. One of the objects illustrated, o_8, is a chair whose centroid is in (13.6, 2.7, 0.6)m. The persistence probability of this chair is low, 0.32, and it was active in the last visit through the environment. This object was visible from poses 18 and 19 and it belongs to supernode 2. An example of pose is given by w_{54}, which is located at (14.5, 6.8)m with respect to the map frame. From this pose, the objects whose keys are 22, 23, 24 and 25 are visible and the poses 53, 55 and 57 are reachable. In addition, the pose belongs to supernode 5. Although a graphical

[2]Object detection is beyond the scope of this paper and out-the-box object detector are going to be used for detecting the different objects found in indoor environments.

representation of the graph is shown in Figure 3.4, the object-based pose graph is stored as a text file including the list of poses and objects and their parameters.

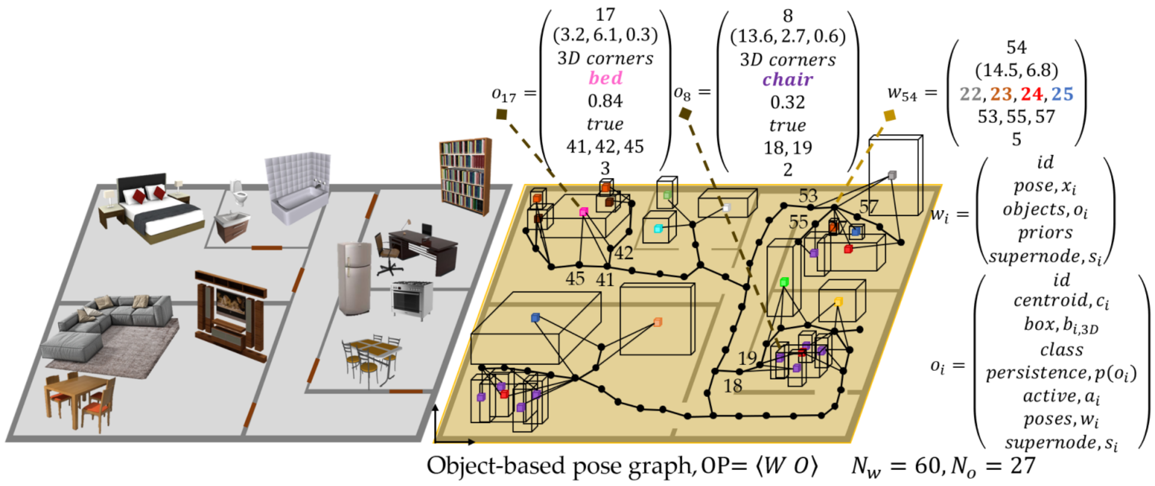


Figure 3.4 Object-based pose graph in an illustrative example of indoor environment. Black dots in the representation correspond to robot poses and cubes corresponds to object. Each object is represented with its centroid (colored cube) and its volume (bigger cubes). For this environment, 60 poses (N_w) and 27 objects (N_o) are created. Information about different poses and objects is given according to the templates in the right. Pose information includes the node unique key, the coordinates of the pose (x-y), the object visible from that pose, the poses connected to that pose and the supernode that the pose belongs to. Object information includes the object key, the coordinates for the object centroid (x, y, z), the volume of the object given by the width, height and length of the bounding box, the object class, its persistence probability and whether it was active in the last mapping session. Objects also include the poses that made them visible and the supernode that the object belongs to.

Objects are the main elements that change in dynamic or non-static indoor environments. For this reason, the structure created to store object information has been prepared to reflect these changes (persistence and active parameters). The object-based pose graph is the component of the hybrid map that will represent the behavior of the environment in the long term and capture its movable elements.

In addition to noticing the changes in the environment, the object-based pose graph allows for learning interesting information about each object and type of object (as explained in Chapter 6). In addition it can improve localization capabilities of the robot and contribute in the semantic knowledge of the environment and semantic-demanding tasks such as object search.

3.5 3D Submaps

The 3D submaps, $M = \{M_1, M_2, ..., M_n\}$, are 3D dense representations of the environment. As mentioned above, a submap is built for every place (room) in the environment. Truncated Signed Distance Fields (TSDFs) are used to build the submaps since they offer a compressed and informative way of representing the environment [122, 178]. The idea behind TSDFs is to represent the environment as a 3D voxel grid in which each voxel contains a sdf value. The sdf is a function that provides the signed distance to the nearest surface for each voxel. The distance will be positive if the voxel is in front of a surface (free area) and negative otherwise (occluded area). Voxels located in a surface will return a distance of 0. In TSDFs, distances are truncated to a maximum value. This simple codification of TSDFs increases its potential compared to occupancy grids as, in addition to knowing which voxels are occupied or free, it knows how far from the occupied region the voxel is. In TSDFs, every voxel is also assigned a weight that indicates how reliable is the sdf value of the voxel. If a voxel is seen several times, the weight increases and the voxel becomes more trustful.

Standard TSDF's voxels, u_i, are defined with their position, $\mathrm{x}_i = (x_i, y_i, z_i)$, their sdf value, sdf_i, and their weight, w_i. In some works, they also include the intensity value, I_i of the voxel in RGB, for the voxels in the surface of obstacles [124]. In the same spirit, in this work, voxels are associated with the object class that the voxel belongs to, $class_i$. This is only known for voxels in the surface of detectable objects, for all the other voxels the object type will be -1. Summarizing, we define a voxel as $u_i = (\mathrm{x}_i, sdf_i, w_i, class_i)$.

As submaps are formed by voxels, each submap contains the information of the voxels that belong to the submap, $\{u_i\}$. In addition, every submap is assigned an identifier that corresponds to the supernode s_i^{id} that the submap represents.

The submaps generated for an illustrative indoor environment are shown in Figure 3.5. Given that the environment consists of 6 rooms, 6 submaps are created. In addition, in Figure 3.6 some of the voxels that form a submap are shown. Voxels lined in white refer to free space and voxels lined with black to surface or occluded regions. Moreover the definition of some individual voxels is shown. For example, u_j corresponds to a voxel in the surface of the object class bed with a certainty of 0.9. In Figure 3.5 and 3.6, voxels are colored according to their height.

Although TSDFs are highly efficient, still they cannot manage operating in large environments. This is one of the reasons why we decided to partition this representation into submaps. Other reasons are the simplification of assigning each submap with a scene type and the possibility to account for the different objects that are in different rooms.

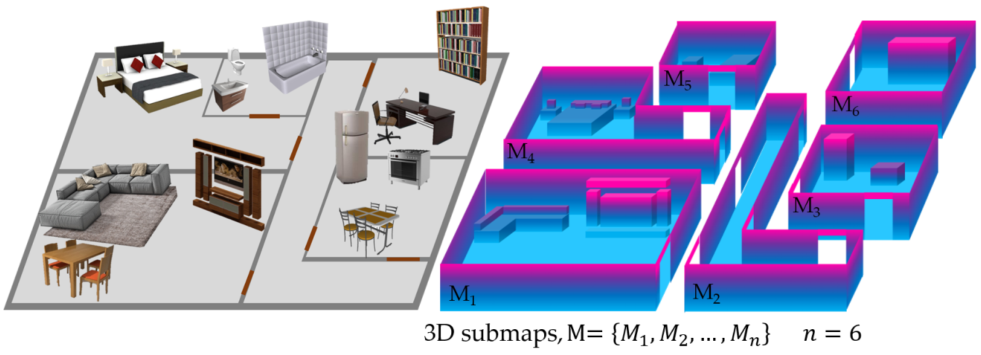


Figure 3.5 3D submaps in an illustrative example of indoor environment. Surface and occluded voxels of the TSDF are coloured while free voxels are not inpainted. For this environment, 6 submaps are generated.

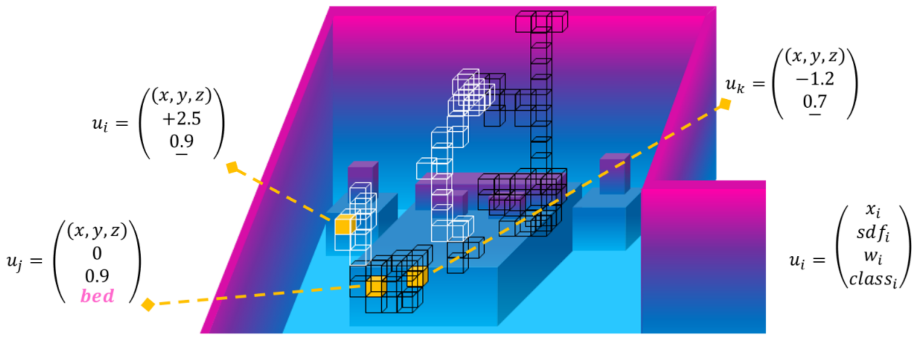

$u_i = \begin{pmatrix}(x,y,z)\\+2.5\\ \underline{0.9}\end{pmatrix}$
$u_k = \begin{pmatrix}(x,y,z)\\-1.2\\ \underline{0.7}\end{pmatrix}$
$u_j = \begin{pmatrix}(x,y,z)\\0\\0.9\\ bed\end{pmatrix}$
$u_i = \begin{pmatrix}x_i\\sdf_i\\w_i\\class_i\end{pmatrix}$

Figure 3.6 Detailed voxel representation for a given submap. Some voxels are highlighted for illustrative purposes. Voxels highlighted in white are free area voxels and voxels highlighted in black are in the surface of objects or occluded. The information contained in three voxels is described according to the template in the right. Only the voxels in the surface are associated with an object class.

3D dense representations include a great amount of detailed information of the environment. Nowadays, almost every approach for autonomous robot operation deals with 3D dense representations as they open up to a wide range of applications. In this work, 3D dense representations are used for localization and path planning. Since it is the representation that contains more information, it is the most suitable for a precise and robust localization. In the case of path-planning, a hybrid method will be presented, that uses the 3D representation to accurately reach the goal. In the future, this representation could be also used for 3D object segmentation and planning paths in the 3D space.

3.6 Component Integration

Once all the components that constitute the hybrid map have been individually explained, we are going to focus on component integration and its sinergies. As a summary, every component serves a different purpose and is in charge of representing some information of the environment. The topological map represents the static structure of the environment and serves as a link between robot and human understanding of the environment. The traversability graph represents the free paths in the environment that the robot can follow to move between different places. This component is used for efficient path planning and exploration. The object-based pose graph represents the position of objects in the environment and the free paths between them. This component serves to identify the movable elements of the environment and manage the navigation in the long term. Finally, the 3D submaps represent the 3D metric information of the environment and serve mainly for accurate localization and path-planning. These purposes are the ones developed in this work, however many more functionalities could be given to each level in the future.

Regarding component integration, the strength of hybrid maps is that different components exchange information in order to make better decisions and increase the efficiency of the whole method. Figure 3.7 shows the connections between the components. Given that it is the most abstract representation, the topological map is the one that exchanges more information with other components. It receives the doors and nodes from the traversability graph and the poses and objects from the object-based pose graph. In addition, it sends the supernodes to all the other components. The object-based pose graph also shares the information of objects with the 3D submaps. In this way, the basic structure on which every component is built (supernodes, nodes, poses and objects) is shared among all the components but stored separately. As shown in the previous sections, these connections are mainly performed using identifiers to match with the actual elements (i. e. the topological map contains the identifiers of the nodes in each room. If more information is needed for a specific node, in the traversability graph, the node with such identifier has to be consulted).

The above explanation directly leads us to analyse how the proposed hybrid map relates to the taxonomy of hybrid maps given at the begining of this chapter proposed by Buschka and Saffiotti [32]. This taxonomy stated that hybrid maps can be classified

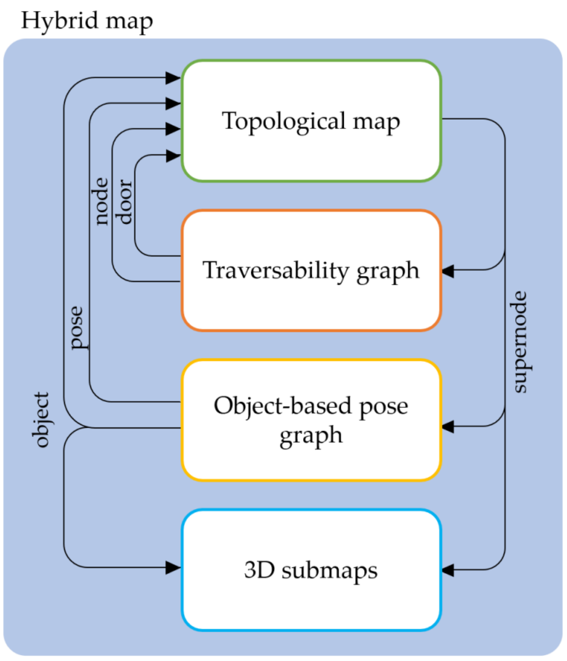


Figure 3.7 Component integration in the hybrid map. The information that each component provides to the others is represented through arrows.

according to three properties: heterogeneity, hierarchy and separability. According to that work, heterogeneity is required and hierarchy and separability are desirable.

The proposed hybrid map is heterogeneous. Our hybrid map is formed by three topological maps and 3D metric submaps, so two different types of representations are distinguishable according to the representation model. In addition, each of the topological maps differs in the type of information contained. The topological map is a toposemantic map in order to understand the environment, and the traversability graph and the object-based pose graph are topometric maps to determine the position of the robot and objects in the environment. Moreover, all the representations differ in the purpose of the information they contain.

The proposed hybrid map is hierarchical. Firstly, it is hierarchical because one of the components represents partially the environment while others represent it fully. In addition, the level of abstraction of the information varies between representations. The reader must have noticed that in this chapter, components were presented from the most abstract one (the topological map) to the most concrete one (the 3D submaps). This hierarchy is also observable in the flow of information between the different components.

The proposed hybrid map is not fully separable. The traversability graph and the object-based pose graph are independent of the other representations and can be built and used on their own (although their functionality might be reduced). On the contrary, the topological map requires information of the traversability graph to be built and the 3D submaps require information from the topological map. If they do not have the required information available, the topological map will identify the whole environment as a single room (which might be erroneous) and a whole 3D map of the environment will be built (which is not recommendable). So, although some components of the hybrid map are separable, the desired performance is only obtained if the components are integrated.

In this chapter the theoretical aspects of the proposed hybrid map have been described and analysed in detail. The remainder of this thesis addresses how the hybrid map (and in detail each of the components) is autonomously built, the strategies to maintain it in long-term operation and how the hybrid map is used for localization and path-planning.

4 Building of the Hybrid Map

Chapter 3 described the theoretical approach for the hybrid map structure that this thesis proposes in order to map indoor environments. The hybrid map consists of several components that need to be built by the robot or provided to it so it can identify its position and the relevant elements to perform autonomous robot navigation. In the luckiest case, the robot will be provided with an a priori map of the environment that contains all the required information. This a priori map could be built by the user or obtained from the abstraction of other representations such as the architect's building layout. However, in most of the cases the robot will not have any prior information of the environment. For this reason, in this chapter, we propose a strategy for building hybrid maps of indoor environments without any prior knowledge. Mapping is the problem of integrating the information gathered with the robot's sensors into a representation and can be described by answering the question "what does the world look like?" [145].

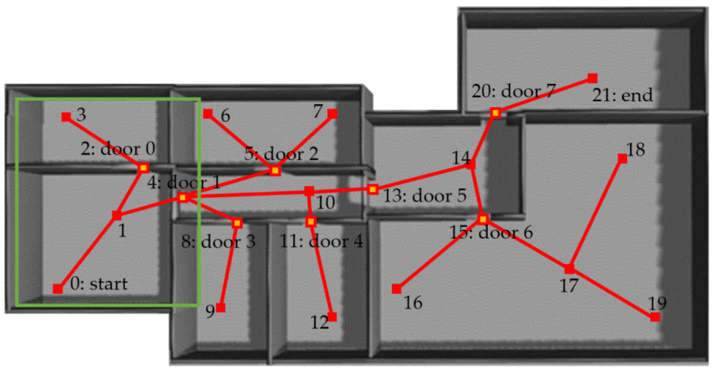
While the robot moves through the environment, each component of the hybrid map will be incrementally built. In addition, the mapping system has to autonomously include the correspondences and links between the different components of the hybrid map. We have designed the mapping process to work concurrently with an autonomous exploration method that will be explained in Chapter 5. Autonomous exploration is performed from a frontier-based perspective in which the system identifies the frontiers, limits between the known area and the unknown one to prioritize the areas to explore. It is and iterative process that visits all the frontiers until the environment is fully discovered. In this chapter, we focus on the construction of the map and we will slightly refer to the exploration algorithm when necessary.

In the following, the method to build each of the components of the hybrid map is explained including the required information for each one and the interconnections between them. Differently from the figures included in Chapter 3, the representations of the maps included in the method explanation of this chapter correspond to the result of mapping the given simulated environment with the described methods. We will firstly describe the method to build the traversability graph since it is the straight-forward representation obtained from the exploration and upon which the other representations are built. We will then explain the construction of the object-based pose graph to finish with the topological map and the 3D metric maps that correspond to each supernode of the topological map. When describing the method to build each map, we will focus on mapping non-cyclic environments but also refer to some specific situations that happen when the robot is facing a loop. We have developed a loop closure algorithm in order to enable the construction of hybrid maps for cyclic environments. After describing the building of each component, we will describe the method for closing loops in the map and we will include an experimental evaluation in non-cyclic and cyclic environments.

4.1 Building of the Traversability Graph

The traversability graph $TG = \langle V, E \rangle$ is a topometric graph of the free space that is built directly using the decision of the exploration method. This representation is built with the navigable areas at the time of exploration and maps the whole environment since the exploration algorithm only finishes when the whole environment has been discovered. The initial pose of the robot is assigned to the first topological node, v_0, of the map and from there the map will grow as the exploration algorithm discovers the environment. The middle point of each visited frontier, m_i, (position that the robot reaches in the exploration strategy) corresponds to a topological node, v_i, registering the number of nodes as N_v. The paths traveled between one node, v_i, and the middle point of the next frontier, m_j, are the topological edges, $e_{i,j}$, accounting the number of edges as N_e.

Every time that the robot arrives to the middle point of a new frontier, m_j, a new topological node is generated, v_j where $j = N_v + 1$, and it is linked to the previous node with the corresponding topological edge. Each edge is annotated with the behavior performed by the robot to reach that node. The performed behaviors will depend on whether the newly created node is semantically labeled as a transit node or a free-area node (as explained later, they are *Move to next free area*, *Approach transit area* and *Cross transit area*). In addition, the semantic information of each node (transit node, free-area node) is stored in the node along with its geometric information (pose) and topological information (connectivity between nodes). The graphical representation of the traversability graph for a simulated indoor environment and the storage representation of the highlighted section of the traversability graph (green box) is shown in Figure 4.1. The list of nodes and edges in the selected region is included. Each node contains information about its identifier, its pose, its type, the list of adjacent nodes and the supernode to whom it belongs. Each edge contains its identifier, the nodes it connects, the behavior for that translation (0 for *Move to next free area* and 1 for *Approach transit area* and *Cross transit area*) and whether it is tranversable or not (true for all the edges at this stage). Note that we create two edges per pair of nodes, one for the movement from i to j and vice versa given that the behaviors to execute may vary.



```
                                e0: nodes 0,1
                                    beh 0
v0: id 0 (0,0)                      tr TRUE
    type 0                      e1: nodes 1,0
    priors 0                        beh 0
    s0                              tr TRUE
v1: id 1 (3, 3.6)               e2: nodes 1,2
    type 0                          beh 1
    priors 0,2,4                    tr TRUE
    s0                          e3: nodes 2,1
v2: id 2 (4.2, 6.1)                 beh 0
    type 1                          tr TRUE
    priors 1,3                  e4: nodes 2,3
    s(-1)                           beh 0
v3: id 3 (0.2, 7.3)                 tr TRUE
    type 0                      e5: nodes 3,2
    priors 2                        beh 1
    s1                              tr TRUE
v4: id 4 (5.6, 4.8)             e6: nodes 1,4
    type 1                          beh 1
    priors 1,5,8,10                 tr TRUE
    s(-1)                       e7: nodes 4,1
                                    beh 0
                                    tr TRUE
```

Figure 4.1 Example of a traversability graph built with exploration in a simulated environment. Lines correspond to edges, red squares to nodes that are classified as free areas and yellow squares to nodes that are classified as transit areas. The list of nodes and edges correspond to the area highlighted in green.

In order to build the traversability graph, the robot must track the current node that the robot is heading, v_j, the last node that the robot visited, v_i, and the total number of nodes, N_v, and edges, N_e, in the map. In this sense, the robot always knows where it comes from, where it is going and the index that will correspond to the next node and edge. This is especially useful in complex situations when the robot may have fully discovered an area in the environment and has to plan a path to continue mapping somewhere else or when the robot reaches a known area from a new path. In the aforementioned cases, keeping track of node and edge information allows to successfully link the new node after path planning to the already-mapped nodes or link the last node that the robot visited with the node that closes the loop when a loop is detected. In the case of path planning, the current node, v_j, and the last node visited by the robot, v_i, will be updated as the path through the known area is executed between initial node for planning, v_{init}, and the node that connects to the unmapped area, v_{goal}. When the new frontier is reached, the new topological node will be created, v_k where $k = N_v + 1$, and it will be linked to the last position of the path, v_{goal}, through a topological edge, $e_{goal,k}$. In the case of loop closure, an original node, v_i, and a looping node, v_{loop}, are identified as the same node by the loop closure strategy (described later in this chapter). The edges that connected v_{loop} to its priors will be changed in order to connect the priors with the original node v_i.

The traversability graph allows the robot to travel safely through all the environment as the nodes are built with the middle position of the frontiers that is the position that maximizes the distance to the obstacles. In further stages, this representation will be used to plan trajectories and move autonomously according to the extracted information.

4.2 Building of the Object-based Pose Graph

The object-based pose graph, $OB = \langle W, O \rangle$, captures the objects and trajectory while the robot explores the environment for the first time. This process implies generating robot poses, w_i, mapping the detected objects, o_i, and connecting poses with objects. In addition, identifying whether the detected objects have been already mapped is required for pose graph consistency.

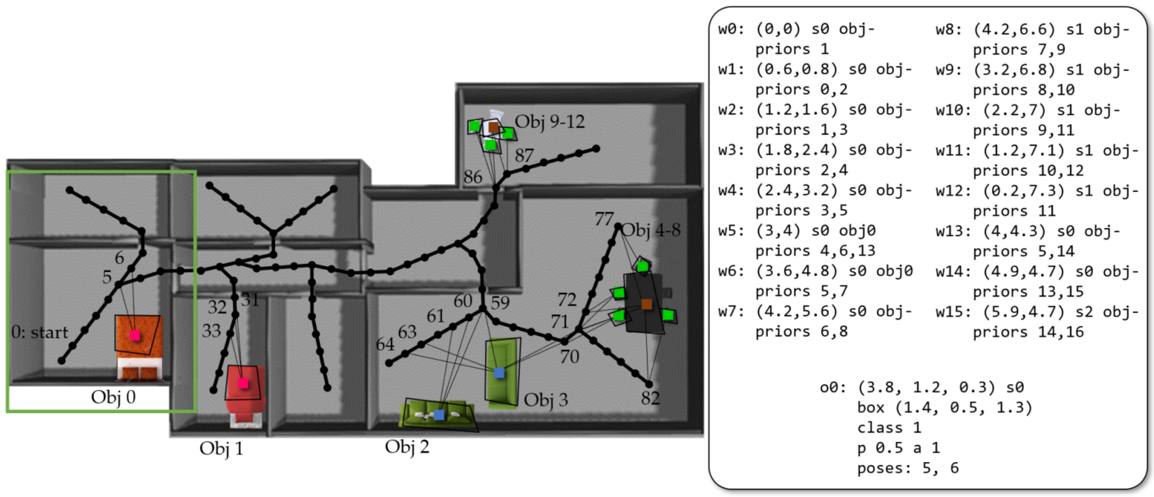
Robot poses, objects and connections are the three main elements of our object-based pose graph (Figure 4.2). Poses are generated according to a fixed traveled distance and connected to its neighbors. In addition, they are connected to the objects that are observable from them and the objects are annotated with the poses from where they were seen. Since poses are generated according to the traversability graph, whenever to robot plans a path or closes a loop the object-based pose graph will behave similarly to the traversability graph. To enable long-term operation, objects are marked according to their persistence probability and active parameter, which are the two attributes that will allow us to manage the presence of objects over time (this will be further described in Chapter 6). When building the object-based pose graph, all the objects will be marked as active and their persistence probability will be initialized according to:

$$p(o_i)_0 = 0.5p(o_i|I_k) \quad , \tag{4.1}$$

where o_i refers to a specific object, $p(o_i)_0$ to the initial persistence probability of object o_i, $p(o_i|I_k)$ refers to the detection confidence of the object detector for a given image frame, I_k, and 0.5 is used as we still can not predict if the object will tend to be static or movable.

Figure 4.2 shows the object-base pose graph in a simulated indoor environment. In addition, it includes the list of poses and objects for the area highlighted in green. Each pose contains its identifier, its 2D position, its supernode, the objects detectable from that pose and the adjacent poses. Each object contains its identifier, the 3D position of its centroid, the dimensions of the 3D box, the object class, the persistence probability and active value for the object and the poses from where the object is observable.

Object detection and representation are two of the main requirements for adding objects to a pose graph. In this thesis, object detection is performed in RGB images through ResNet-101 [75] trained with COCO Dataset [105]. Detections contain the objects, o_i, their detection confidence, $p(o_i|I_k)$, and the 2D bounding boxes of the objects, $b_{i,2D}$, for that given image frame, I_k, in the time step k. Regarding object representation, the most common approaches for pose graphs are object reconstruction [113] and cuboid generation [175]. Both methods require an object detector that provides the information of the object position and class. With this information, reconstruction approaches group the 3D points that belong to an object and extract the object geometry; on the contrary, cuboid generation approaches identify the 3D bounding box of the object representing the space that it occupies. Regarding limitations, object reconstruction is highly demanding for memory and processing requirements but gives an individual and detailed representation of objects. Cuboid generation overlooks the specific characteristics of each object instance within the class with the advantage of reducing the computation cost. In this work, a cuboid generation method is proposed



```
w0: (0,0) s0 obj-         w8: (4.2,6.6) s1 obj-
    priors 1                  priors 7,9
w1: (0.6,0.8) s0 obj-     w9: (3.2,6.8) s1 obj-
    priors 0,2                priors 8,10
w2: (1.2,1.6) s0 obj-     w10: (2.2,7) s1 obj-
    priors 1,3                priors 9,11
w3: (1.8,2.4) s0 obj-     w11: (1.2,7.1) s1 obj-
    priors 2,4                priors 10,12
w4: (2.4,3.2) s0 obj-     w12: (0.2,7.3) s1 obj-
    priors 3,5                priors 11
w5: (3,4) s0 obj0         w13: (4,4.3) s0 obj-
    priors 4,6,13             priors 5,14
w6: (3.6,4.8) s0 obj0     w14: (4.9,4.7) s0 obj-
    priors 5,7                priors 13,15
w7: (4.2,5.6) s0 obj-     w15: (5.9,4.7) s2 obj-
    priors 6,8                priors 14,16


            o0: (3.8, 1.2, 0.3) s0
                box (1.4, 0.5, 1.3)
                class 1
                p 0.5 a 1
                poses: 5, 6
```

Figure 4.2 Example of an object-based pose graph built with exploration in a simulated environment. Black lines and dots represent the path traveled by the robot and the poses generated. Each object is represented with its 3D bounding box (cuboid) and its centroid. Different centroid colors refer to different object classes. In addition, the centroid of the objects are connected to the poses where the object was observed. The list of poses and objects correspond to the area highlighted in green.

because we are interested in classifying objects according to their class but we do not require a detailed description of the object.

As the object detector used provides the 2D bounding box of each object detected, we use 2D bounding boxes and point cloud information to calculate the centroids, $c_i = [x_i, y_i, z_i]$, and vertices of the 3D bounding boxes (cuboids), $b_{i,3D} = [w_i, h_i, d_i]$, where the object, centroid and bounding box are linked through their indexes. Only objects with high detection confidence, $p(o_i|I_k) > 0.7$, are included in the map. Since the details of the object detection algorithm are out of the scope of this thesis, we consider the recognitions over that confidence value reliable without further parameter tuning. Object detection in a simulated environment is shown in Figure 4.3 (a) and cuboid generation is shown in Figure 4.3 (b). There are two cuboids because two instances of the object were detected.

Point cloud information is used to calculate transformations between image pixels and coordinates in the map, as presented in [76]. Such transformations are firstly used to obtain the object depth. Object depth is characterized using its minimum, mean and maximum values. Minimum and maximum values are calculated for defining vertex depth and mean value for centroid depth. Minimum object depth is calculated as the median of the 2% smallest depths within the 2D bounding box of the object. Maximum object depth is calculated through an adaptation of the flood fill algorithm, starting from the minimum-object-depth pixel and visiting its neighbors. If the two pixels are connected (difference in depth smaller than a threshold, θ), the pixel is considered to belong to the object. On the contrary, it is considered part of the background and it is discarded. The maximum object depth is the maximum value within the depths that belong to the object. Mean object depth is calculated as the mean value between minimum and maximum object depths.

3D coordinates of the object centroid are obtained using the mean object depth and pixel-to-coordinate transformation. Similarly, 3D coordinates of the bounding box vertices are estimated using the minimum and maximum object depths and the width and height of the 2D bounding box.

This method provides a representation of every object that the robot sees using a 3D cuboid characterized by its vertices and its centroid from RGB images and point cloud information. However, we do not want to map all the objects detected as objects could be mapped several times if they are seen in multiple occasions. For this reason, object merging is the third requirement for adding objects to a pose graph.

Object merging involves identifying that an object detected in image frame, I_l, in time step l was previously seen from the same or another position in image frame, I_k where $l > k$. In this work, every object to be added to the map is analyzed according to its class and position. Firstly, two objects are only going to be merged if they belong to the same object class. Secondly, two conditions are introduced to evaluate the relation between object positions:

- In first place, the centroid of the new object is evaluated in order to check whether it lies inside the cuboid of the already-mapped object, $c_j = (x_j, y_j, z_j) \in b_{i,3D}$. If it happens so, they are automatically merged without any modification.

- If the previous condition is not met, the area of influence, $a_{inf}(o_i)$, of the already-mapped object is calculated. The area of influence of an object is defined as a sphere centered in the object centroid where, given the size of the object, no other centroid of the same object class can be found, eq. 4.2. This sphere is defined by the diagonal of the object $\sqrt{b_{i,3D}^2}$ multiplied by a tolerance factor α, since objects from the same object class may have approximately similar sizes but not exactly the same. Once the area of influence for the already-mapped object is calculated, if the centroid of the new object lies within this area, the two objects are merged and the cuboid of the object is reshaped.

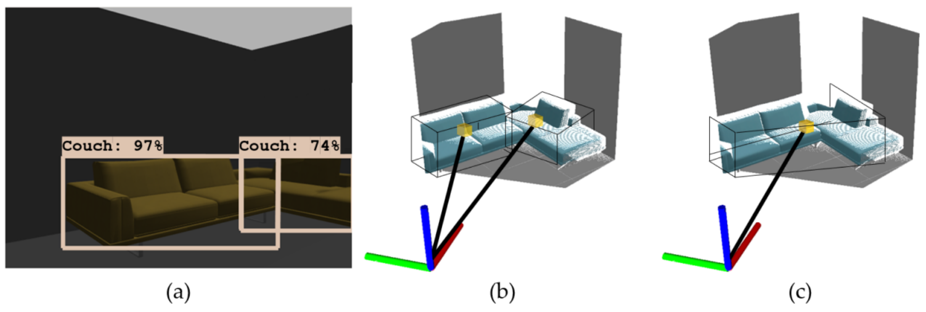


Figure 4.3 Object detection and generation of object cuboid. In (a), the frame, class and confidence of a detected object is shown. In (b), the cuboid and centroid for the object is calculated using the frame of the object and the current point cloud. As the centroid of the second detection (right) is within the area of influence of the first detection, in (c) both instances are merged.

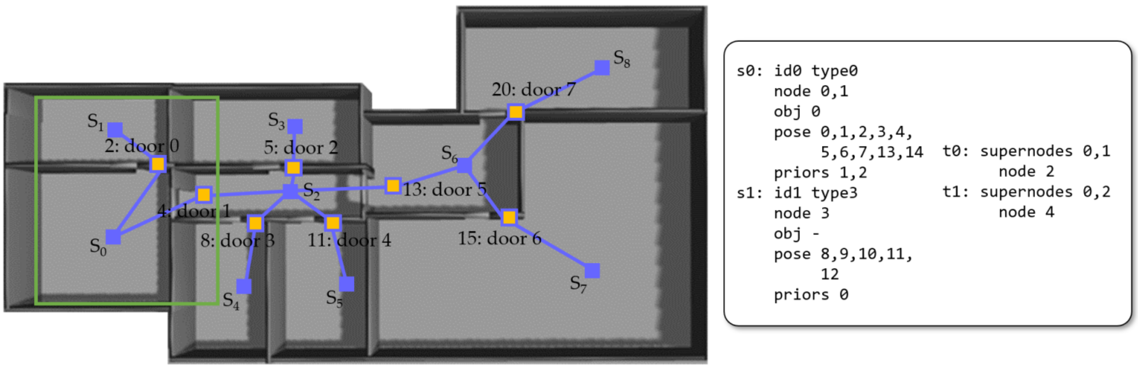
$$a_{inf}(o_i) = \alpha\sqrt{b^2_{i,3D}} \quad . \tag{4.2}$$

If the centroid lays in the area of influence, $c_j \in a_{inf}(o_i)$, the cuboid of the already-mapped object is reshaped to include the centroid of the new detection. This situation is illustrated in Figure 4.3 (c).

The described object detection, object reconstruction and object merging strategy allows to build a consistent object-based pose graph of the environment.

4.3 Building of the Topological Map

The topological map $TM = \langle S, T \rangle$ is a toposemantic representation of the structure of the environment in which nodes are related to rooms. For the seek of clarity, the nodes of the topological map are called supernodes, s_i, given that they group nodes from other more detailed layers of the hybrid map. Specifically, they group the nodes of the traversability graph, v_i, and the poses, w_i, and objects, o_i, of the object-based pose graph that belong to a given room. Each transition between supernodes, $t^{[k]}_{i,j}$, is identified with the transit node from the traversability graph, v_l, that connects the two rooms i and j. Transit nodes are also denoted as doors, where each door $d^{[l]}_k$ links the transition k with the topological node l. In this way, the topological map establishes the connectivity between rooms in the environment, as shown in Figure 4.4. In addition to a graphical representation of the topological map, Figure 4.4 shows the list of supernodes and transitions of the highlighted region of the environment. Supernodes



```
s0: id0 type0
    node 0,1
    obj 0
    pose 0,1,2,3,4,
         5,6,7,13,14  t0: supernodes 0,1
    priors 1,2              node 2
s1: id1 type3         t1: supernodes 0,2
    node 3                  node 4
    obj -
    pose 8,9,10,11,
         12
    priors 0
```

Figure 4.4 Example of a topological map built with exploration in a simulated environment. Lines correspond to edges, purple squares to supernodes and yellow squares to transitions. The list of supernodes and transitions correspond to the area highlighted in green.

contain information about their identifiers, the type of room, the nodes, objects and poses that belong to the supernode and the adjacent supernodes. Transitions contain information about their identifier, the supernodes they connect and the topological node linked to the transition.

In order to build the topological map, the robot starts by assuming that it is in the initial supernode, s_0, until a transit area node, $t_{0,1}^{[0]}$, is detected, checked and crossed. That is the moment in which the first connection of the topological map is created and the supernode is updated, s_i where $i = N_s + 1$ and N_s is the total number of supernodes. As the traversability graph maps the nodes and tracks the current node, the topological map assigns the supernodes to each room in the environment and tracks the current supernode and maximum supernode, $s_{max} = N_s$. In addition, a similar process to the one described for the traversability graph takes place when dealing with path planning and loop closure.

Regarding path planning, if the robot is planning a path through a known area that leads to a new frontier to visit in a different room, it must keep track of the current supernode in order to successfully connect the new discovered room to its adjacent known room. For example, in Figure 4.4, the robot might have completely visited supernode 1 (s_1) and in order to continue exploring the environment it has to return to supernode 0 (s_0) to then start mapping supernode 2 (s_2). To successfully represent this sequence, the robot needs to track the change to supernode 0 and correctly establish the transition between supernode 0 and supernode 1, $t_{0,1}^{[0]}$, and supernode 0 and supernode 2, $t_{0,2}^{[1]}$.

Regarding loop closure, if the robot was in a room, s_j, and it enters an already-mapped room from another path, it will first think it is a new room (until the loop closure strategy confirms that it is an already-mapped room) and it will assign a new supernode identifier to the room, s_k where $k = N_s + 1$. When the loop is confirmed, the information of the original room, s_i, and the loop room, s_k, will be merged under the original supernode, s_i. Also the new transition to the supernode will be added to the original supernode, $t_{j,i}^{[l]}$.

The topological map corresponds to the stable structure of the environment as rooms and transitions between them are not very likely to change in standard indoor environments.

4.4 Building of the 3D Submaps

In the proposed hybrid mapping method, a 3D submap is built for every room in the environment. Two 3D mapping systems have been used in this thesis, Voxblox [122] and TSDF Fusion[178]. In this first part of the thesis, we will be using Voxblox and TSDF Fusion will be used for the metric localization method presented in Chapter 7. Voxblox is mainly meant to build Euclidean Signed Distance Fields (ESDFs) out of Truncated Signed Distance Fields (TSDFs). TSDFs are built from point clouds while the robot moves in the environment and are included in the ESDF global map. ESDFs give Euclidean distance to the nearest obstacle at any point in the map which makes them suitable for mobile robot tasks.

In order to build the 3D submaps, Voxblox is integrated with a method to partition 3D maps according to the topological map. The topological map and the 3D submaps are built simultaneously in real-time. Submaps, $M = \{M_1, M_2, ..., M_n\}$, are generated according to the traversability of doors, $D = \{d_1^a, d_2^b, ..., d_m^c\}$, as every time the robot crosses a door, it enters a new supernode and, thus, it must start mapping a new submap.

In order to build the 3D submaps, Voxblox is launched to start building submap M_0. When crossing a new door, $d_i^a \notin D$, the current submap, M_i, is saved and a new one starts to build, M_j where $j = N + 1$. When crossing a previously crossed door, $d_i^a \in D$, that connects the current submap, M_i, to another submap, M_j, current submap is saved, M_i, previously built submap loads, M_j, and continues building it. The task of saving and loading the maps is performed through Voxblox methods. As a result, the environment is partitioned according to its rooms without the need of post-processing the map. The number of submaps generated will correspond to the number of rooms in the environment. Each submap will be associated to a supernode and the transit area nodes that lead to it. Similarly to what happened with the topological map when a loop is found, current submap, M_{loop}, and original submap, M_j, are merged and stored in M_j. The method proposed in combination with Voxblox outputs a dense 3D submap for each of the rooms of the environment as shown in Figure 4.5.

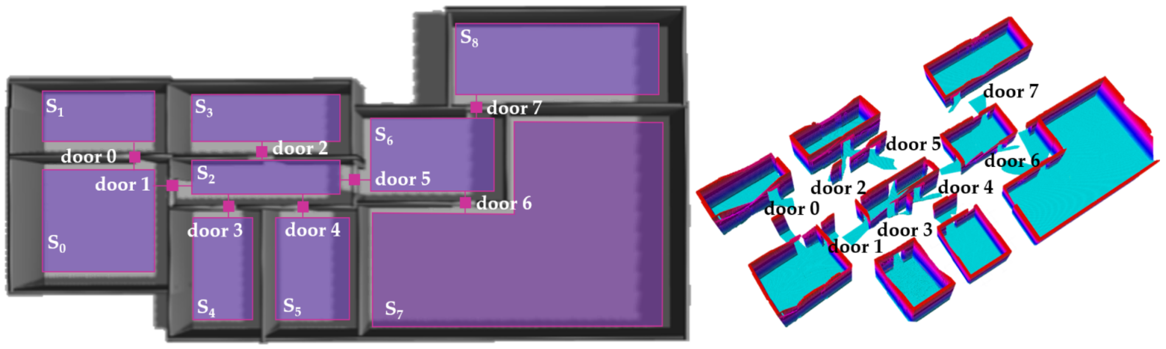


Figure 4.5 Example of a the submaps built with exploration in a simulated environment. A submap is built for every room and the correspondence of the environment and the resulting submaps can be observed with the annotated doors.

During the description of all the map structures, we have been referring to how the maps will behave when encountering a looping situation. Prior to evaluating the described methods with simulated and real-world experiments, the loop closure method is presented.

4.5 Loop Closure for Cyclic Environments

The hybrid mapping strategy described up to now is able to map indoor non-cyclic environments but a loop closure algorithm is required to map cyclic environments. We have already mentioned how each component will behave when a loop is detected. However, we have not presented the method to identify a possible loop in the environment. Since our robots will have to face loops or cyclic situations (i. e. kitchens with two entrances or complex office environments), loop closure algorithms must be provided to them. An example of looping situation is shown in Figure 4.6 (right).

Loop closure has been widely addressed by researchers. Most of SLAM approaches only considered geometric conditions to perform loop closure [60] although some authors added visual features to the geometric information in order to have more robust results [97, 118]. Other authors have tackled loop closure through localization. In [156], a probability distribution is maintained during the exploration and whenever there are two peaks in the distribution (two nodes with high probabilities) the algorithm tracks those nodes to look for a convergence, loop, or divergence, different locations. Another approach, [157], is maintaining a tree with every possible hypothesis. Each hypothesis is associated with its probability and the tree is pruned until a decision is taken.

We propose to use the geometric, topological and semantic information available for the nodes of the traversability graph to estimate the uncertainty of being in a loop. We consider that the robot is in a loop when it is visiting (or closely visiting) an area that it has already visited. According to the proposed algorithm, this situation only happens when an area is reachable from two different paths from the same starting position. For the seek of clarity, if the robot is visiting an area from the same path it is because the robot is performing path planning, so it is already aware that the areas visited are known. The process for identifying and accepting a loop is described in Figure 4.6 (left). The robot will check the semantic, geometric and topological correspondences in a staged fashion.

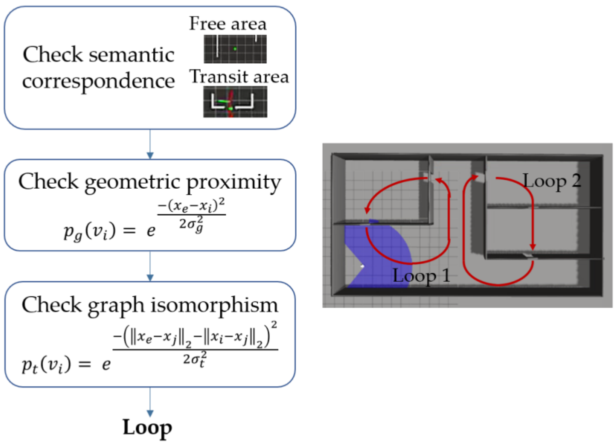

Check semantic correspondence
Free area
Transit area
Check geometric proximity
$p_g(v_i) = e^{\frac{-(x_e - x_i)^2}{2\sigma_g^2}}$
Check graph isomorphism
$p_t(v_i) = e^{\frac{-\left(\|x_e - x_j\|_2 - \|x_i - x_j\|_2\right)^2}{2\sigma_t^2}}$
Loop
Loop 2
Loop 1

Figure 4.6 Loop closure strategy and example of a simulated environment with two loop situations (one connecting two rooms and the other one connecting three rooms). The loop closure strategy first evaluates the semantic correspondence, afterwards the geometry proximity and finally the topological similarity through graph isomorphism.

In this thesis, loop probability is built from geometric, semantic and topological information. Firstly, semantic information is considered as the condition for the original node and the loop node of belonging to the same semantic type (free area or transit area). If this condition is met, the geometric proximity is evaluated. Geometric proximity is considered as the difference between the expected position of a node and the current position of the possible loop node. Geometric loop probability for a node v_i, $p_g(v_i)$, has been computed using a Gaussian distribution, according to eq. (4.3) and it determines how likely it is that the robot is geometrically close to an already known area. Terms x_e and x_i refer to the expected and loop positions of the node, respectively, and σ_g refers to a variance value that has been experimentally tuned.

$$p_g(v_i) = e^{\frac{-(\mathrm{x}_e - \mathrm{x}_i)^2}{2\sigma_g^2}} \quad . \tag{4.3}$$

In the case that two nodes of the same semantic type obtained a geometric loop probability greater than a threshold (experimentally set to 0.3), a process to check the topological correspondence starts. This is performed through graph isomorphism over the traversability graph. Graph isomorphism consists in replicating the morphology of the graph associated to the original node starting from the possible loop node. In other words, once the robot has identified the original node, it will check whether it is possible to travel the same path that it traveled the first time that it visited that node. In this process, the first step is to verify that a prior of the original node is reachable from the loop node. A prior is considered reachable when there is not any obstacle between the current position and the prior position. If it is reachable, the robot moves until it reaches the prior node. The expected distance and the loop traveled distance are compared to determine the topological loop probability $p_t(v_i)$, eq. (4.4). Terms x_e, x_i and x_j refer to the expected, loop and prior position of the nodes, respectively, and σ_t refers to a variance value that has been experimentally tuned.

$$p_t(v_i) = e^{\frac{-(\|\mathrm{x}_e - \mathrm{x}_j\|_2 - \|\mathrm{x}_i - \mathrm{x}_j\|_2)^2}{2\sigma_t^2}} \quad . \tag{4.4}$$

This process iterates and compares the graphs until a positive or negative decision is reached according to the global loop probability, $p_{loop}(v_i)$. Normalized global loop probability, eq. (4.5), relates geometric and topological uncertainties. Index k refers to the number of graph isomorphism iterations.

$$p_{loop}(v_i) = \eta p_g(v_i) \prod_k (p_t(v_i)_k) \quad . \tag{4.5}$$

A positive decision implies the acceptance of the loop and the update of the maps to this new situation. A positive decision is reached when the global probability surpasses a threshold value (set to 0.9, high certainty that the robot has detected a loop) within five iterations of graph comparison. A negative decision implies that the loop is rejected and the maps are not modified. A negative decision is reached when the global probability does not surpass the threshold value within five iterations, or when any of the priors is not reachable from the loop node or any of the nodes generated during the graph isomorphism process.

Once the methods for building of the hybrid map and loop closure are explained, we can proceed with their experimental evaluation. Different simulation and real-world experiments will show the results for building the different layers of the hybrid map in non-cyclic and cyclic environments.

4.6 Experimental Evaluation

An incremental evaluation of the map building methods is presented according to the different maps that constitute the hybrid map for cyclic and non-cyclic environments. Firstly, we include the construction of the traversability graph and the topological map in Section 4.6.2. In Section 4.6.3, also the 3D submaps are built and we compare to other submapping strategies. Section 4.6.4 includes an experiment to build the object-based pose graph in a real indoor environment. Finally, in Section 4.6.5, the whole hybrid mapping system is evaluated through simulated and real-world experiments.

4.6.1 Experimental Setup

Our mapping methods have been evaluated both in simulated and real-world environments. The robot was autonomously moving through the environment for all the simulated environments and also the real-world environments except for the object-based pose graph experiment, in which the robot was teleoperated. A frontier-based exploration method was used to drive the robot autonomously. This algorithm is explained in Chapter 5.

Mapping experiments were conducted firstly using Gazebo simulation environment. The performance and resulting maps were visualized using RViz since the work was developed using ROS (Robot Operating System framework) and C++. Several indoor environments were created to develop the experiments as close as possible to real situations. The simulated version of Turtlebot 2 robot is equipped with the simulated sensors Hokuyo URG-04LX laser sensor and Asus Xtion Pro camera to perceive the environment. Regarding the computational resources, for both simulated and real-world experiments, a PC with IntelCore i7-6500U CPU@2.50Hz 12GB RAM was used.

Real-world experiments in different indoor environments (a corridor area and a meeting room in University Carlos III of Madrid and several houses) are also presented. Turtlebot 2 robot was used for the experiments and RViz visualizer was used to observe the resulting map of the environment. As in the simulated experiments, Turtlebot 2 is equipped with a Hokuyo URG-04LX laser sensor and an Asus Xtion Pro camera.

All the experiments presented in this thesis involving the object-based pose graph were performed with the same values for the depth threshold, θ and tolerance factor for merging, α. The value of θ was set to 15 cm and α to 0.9. We have empirically evaluated that these choices are suitable for different objects present in the environment and for different environments, see Table 4.1. The first three columns in Table 4.1 refer to θ: *mean diff.* represents the average difference between depth of neighboring pixels for each object class (cm); *% DIFF* refers to the percentage of differences that are greater than 15 cm; and *% Error* refers to the final error in size comparing the real and calculated depth for the object. Although several objects have differences between individual pixels higher than the defined threshold, this only affects the chairs with a 5.91% error (a 50 cm-wide object will be detected as 47.05 cm wide). Next three columns refer to α: *separation* column shows the minimum separation (m) between objects of the same class present in any of the environments; *Diagonal* column shows the maximum diagonal value (m) for each of the object classes; and finally, *Diagonal** refers to the area of influence of the object which corresponds to 0.9 times the maximum diagonal. Although the maximum diagonal values for some objects are larger than the minimum distance (what would lead to an error), using 0.9 as α value solves these possible errors.

Table 4.1 Evaluation of parameters θ (depth threshold) and α (tolerance factor for merging).

object	θ			α		
	mean diff.	% DIFF	% Error	separation	Diagonal	Diagonal*
chair	1.62	1.86	5.91	0.67	0.69	0.62
sofa	1.23	1.06	0	1.85	1.89	1.70
plant	1.57	1.41	0	-	0.63	0.57
cup	0.41	0	0	0.44	0.15	0.13
bottle	0.42	0	0	1.54	0.09	0.08
laptop	0.35	0	0	-	0.46	0.41

4.6.2 Evaluation of the Topological Map and Traversability Graph

This evaluation includes simulated experiments in medium-size house-like environments, a big office environment and a real-world house environment.

Evaluation in Simulated Medium-size House-like Environments

The construction of the topological map and the traversability graph was tested in two different non-cyclic simulated house-like environments, one medium-size house environment (130.5m^2) and one big-size house environment (235m^2) and in a cyclic indoor environment (342m^2). The cyclic environment has two loops, one connecting two rooms together and the other one closing a loop that traverses three rooms. Figure 4.7 contains the traversability graphs from different starting positions for the three environments. In addition, Figure 4.8 contains the information of the topological

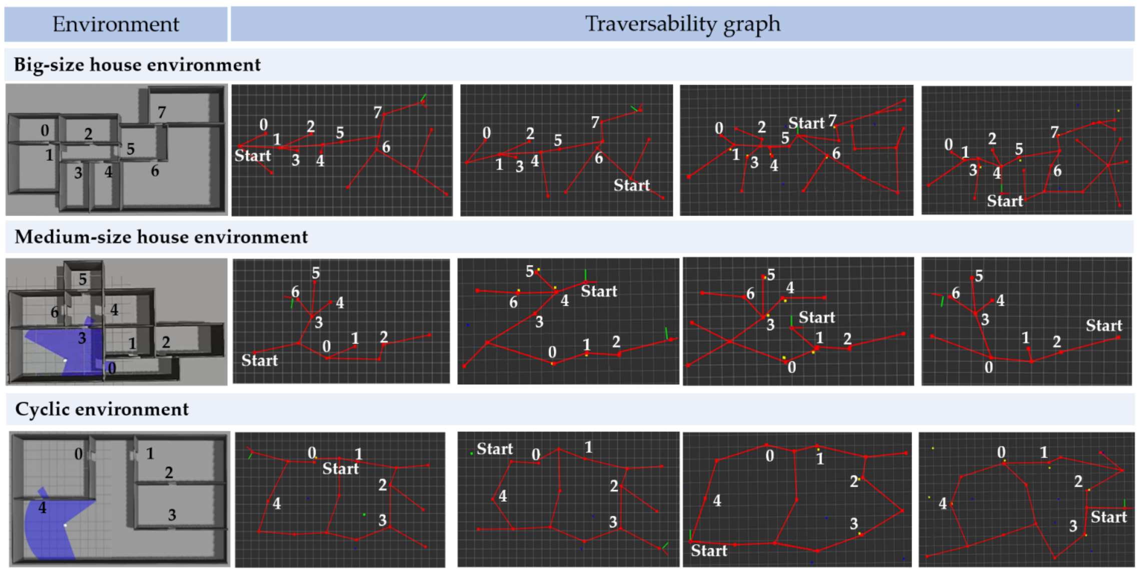
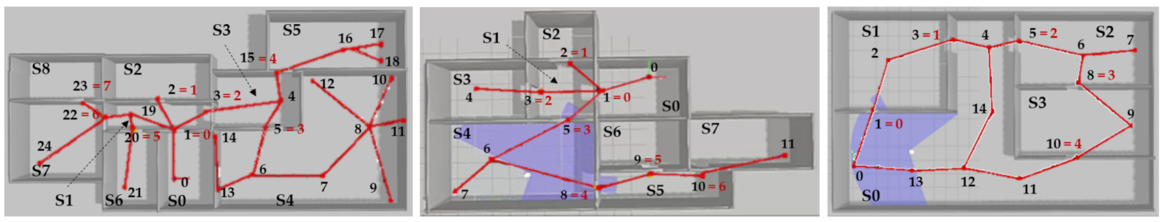


Figure 4.7 Result for simulated non-cyclic and cyclic indoor environment. The first row shows the big-size non-cyclic simulated environment and the traversability graphs obtained from different starting positions. The second row shows the medium-size non-cyclic simulated environment and the traversability graphs obtained from different starting positions. The third row shows the cyclic indoor simulated environment and the traversability graphs obtained. Correspondences between doors and starting positions are included to facilitate the interpretation of the graph structures.

map for one of the traversability graphs shown in Figure 4.7 per environment. The list of the supernodes and transitions that constitutes the topological map is also included.

The big-size house environment and the resulting traversability graph from the exploration are shown in Figure 4.7 (first row). Four resulting traversability graphs from initial random positions are shown. A similar structure is observable in the graphs and the tested environment. Differences are due to the difference in the initial position. In all of the cases, the eight doors were detected and the whole environment was mapped. The topological map is also successfully built for this environment, an example that corresponds to the fourth traversability graph is listed in Figure 4.8 (a).

Regarding the experiments for the medium-size house environment, results are shown in Figure 4.7 (second row). Four resulting traversability graphs from initial random positions are shown. All the rooms were mapped in the four cases and the



S0: Nodes: 0
S1: Nodes: 19
S2: Nodes: -
S3: Nodes: 4
S4: Nodes: [6, 14]
S5: Nodes: [16, 18]
S6: Nodes: 21
S7: Nodes: 24
S8: Nodes: -

Total supernodes: 9

Transition 0: (0, 1) Node: 1
Transition 1: (1, 2) Node: 2
Transition 2: (1, 3) Node: 3
Transition 3: (3, 4) Node: 5
Transition 4: (3, 5) Node: 15
Transition 5: (1, 6) Node: 20
Transition 6: (1, 7) Node: 22
Transition 7: (7, 8) Node: 23

Total transitions: 8

(a)

S0: Nodes: 0
S1: Nodes: -
S2: Nodes: -
S3: Nodes: 4
S4: Nodes: 6, 7
S5: Nodes: -
S6: Nodes: -
S7: Nodes: 11

Total supernodes: 8

Transition 0: (0, 1) Node: 1
Transition 1: (1, 2) Node: 2
Transition 2: (1, 3) Node: 3
Transition 3: (1, 4) Node: 5
Transition 4: (4, 5) Node: 8
Transition 5: (5, 6) Node: 9
Transition 6: (5, 7) Node: 10

Total transitions: 7

(b)

S0: Nodes: 0, 4, 11, 12, 13, 14
S1: Nodes: 2
S2: Nodes: 6, 7
S3: Nodes: 9

Total supernodes: 4

Transition 0: (0, 1) Node: 1
Transition 1: (1, 0) Node: 3
Transition 2: (0, 2) Node: 5
Transition 3: (2, 3) Node: 8
Transition 4: (3, 0) Node: 10

Total transitions: 5

(c)

Figure 4.8 Example of topological maps (characterized by the supernodes and trasitions) for the shown traversability graphs. One example is included for each of the house-like environments in (a), (b) and (c). Images show the traversability graph annotated with its nodes (black), the supernode assigned to each room and the identifier of the transition that connects each supernode (red). Lists show each supernode along with the nodes that belong to it and the transitions that connect two supernodes (X, Y). In addition, the node of the traversability graph that corresponds to the transition is included.

influence of initial position in the resulting maps is minimal. Although different starting positions result in different graph structures, the obtained traversability graphs faithfully represent the environment. The topological map is successfully built for all the initial positions, an example that corresponds to the second traversability graph is listed in Figure 4.8 (b).

Experiments that evaluate the map building and the loop closure algorithm are included through a simulated cyclic environment, Figure 4.7 (third row). Traversability graphs were created from different starting positions; all of them successfully mapped the environment and closed the two loops. The two loops are observable in the resulting graphs where the robot mapped the whole environment and detected all the doors. The loop connecting door 1 and door 5 was successfully closed after 2 iterations of graph isomorphism and the loop connecting doors 2-4 was successfully closed after 3.75 iterations (average values). The topological map that corresponds to the third traversability graph is listed in Figure 4.8 (c).

Evaluation in a Simulated Big Office Environment

The construction of the traversability graph and the topological map was also tested in a more complex and bigger simulated office environment. This environment (1141.26m^2) is composed of 24 rooms of very different sizes and includes four short loops (connecting 2 or 3 rooms together). The environment is shown in Figure 4.9 (a) and the resulting traversability graph in Figure 4.9 (b). The robot mapped the whole environment, visited all the rooms and closed the four loops. The traversability graph consists of 67 nodes and 69 edges. 27 nodes were recognized as transit areas and 40 as free areas.

Evaluation in a Real-world Environment

Experiments in a house were developed using Turtlebot 2 robot to verify the viability of the proposed method in the real world. Experiments were performed in an empty house of approximately 60m^2 consisting of five rooms (four rooms and a corridor). In Figure 4.10 (a) the robot is shown in the real house environment and Figure 4.10 (b) shows the resulting map of the environment. A schematic view of the environment is shown along with the generated graph. The graph is built with 15 nodes (four of them classified as transit areas) and 14 edges connecting them. The nodes identified as transit areas divide the supernodes of the topological map, since 4 nodes were identified as transit areas 5 supernodes were mapped that correspond to the rooms in the environment.

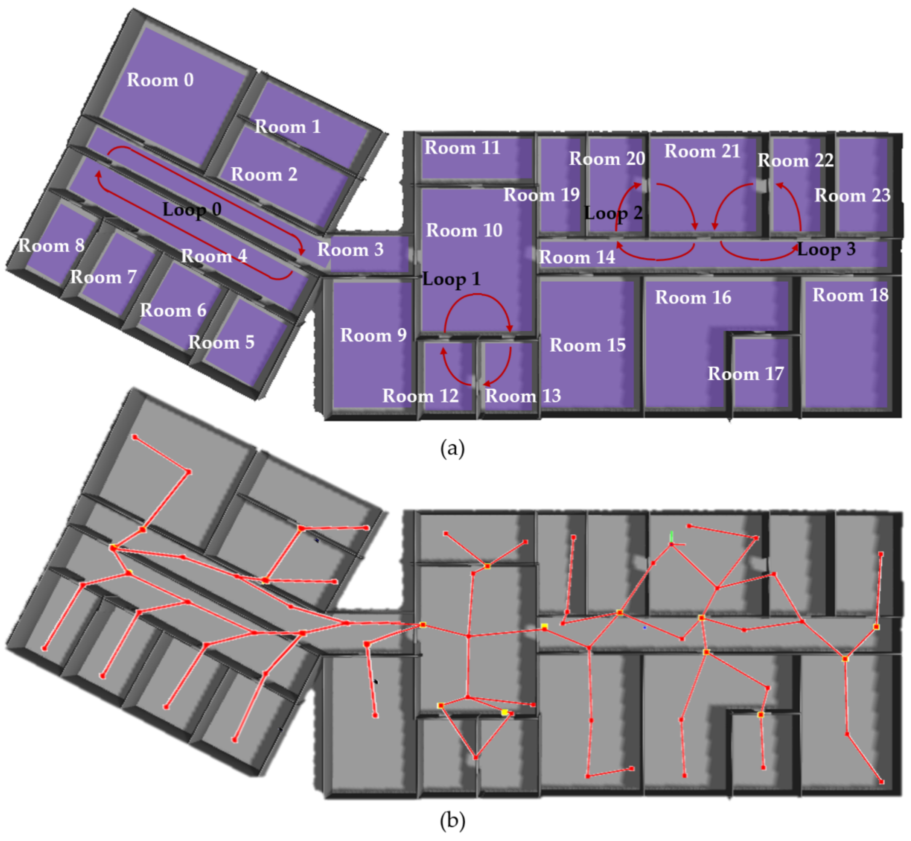


Figure 4.9 Results building the traversability graph for a big office environment. The environment and the different rooms and loops are indicated in (a); the resulting traversability graph is shown in (b).

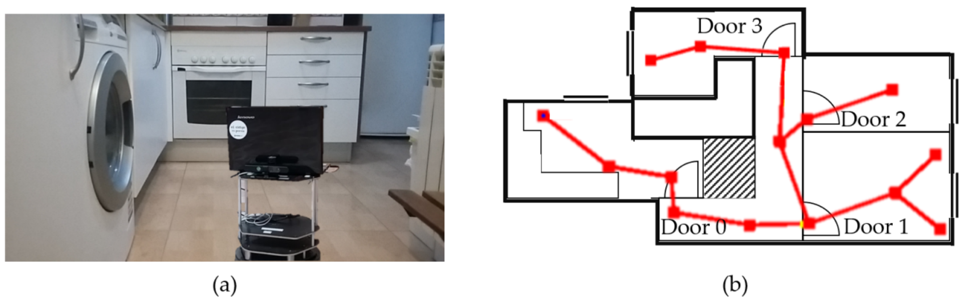


Figure 4.10 Real topological mapping experiment in a house environment. A picture of Turtlebot 2 robot during the mapping process is shown in (a). In (b), we show the traversability graph of the real house environment. The traversability graph consists of 15 nodes (4 of them classified as transit area and 11 classified as free area) and 14 edges.

4.6.3 Evaluation of the Topological Map, the Traversability Graph and 3D Submaps

This evaluation includes mapping a simulated big office environment, the comparison to fixed submapping of the environment and evaluation in a real-world hallway area of University Carlos III of Madrid.

Evaluation in Simulated Environments

The first validation regarding the construction of the 3D submaps is to test the improvement in memory requirements when mapping an indoor environment. RAM resources used when building the topological map, the traversability graph and 3D metric submaps (in the following, topological-submap representation) are compared to when building only a global 3D map of the environment. Experiments were conducted in a simulated big office environment of 1137.5m^2 (shown in Figure 4.12 (a)) and map resolution (voxel size) was set in both cases to 5cm. The environment consists of 22 rooms and 4 loops (3 simple loops connecting together 2 or 3 rooms and 1 complex loop).

The resulting representations for global 3D map and topological-submap representation are shown in Figure 4.12 (b) and (c), respectively. For the topological-submap representation, the traversability graph and the submaps built in the environment are shown. The comparison in the increment in RAM resources during mapping is included in Figure 4.11. Grey line corresponds to global 3D map and purple line corresponds to topological-submap representation. Final memory value with topological-submap representation is 3.2 times smaller than with global 3D map. This is due to the contin-

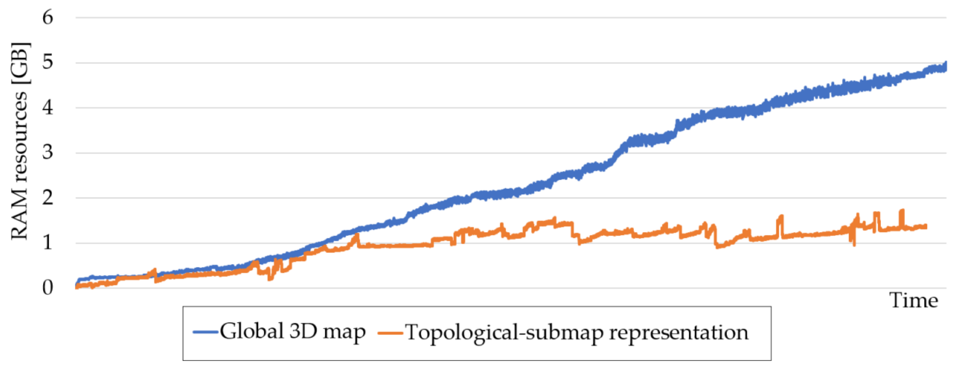


Figure 4.11 Comparison in RAM resources required when building global 3D map of the environment and building the topological-submap representation.

uous mapping for global map, whereas for submapping, the map is continuously saved and reset maintaining a smaller merging area (resulting in small peaks on memory consumption). It is worth mentioning that the resources requirement for global 3D mapping grows continuously whereas for topological-submap representation a steady behavior is obtained as its upper bound is set by the largest room in the environment.

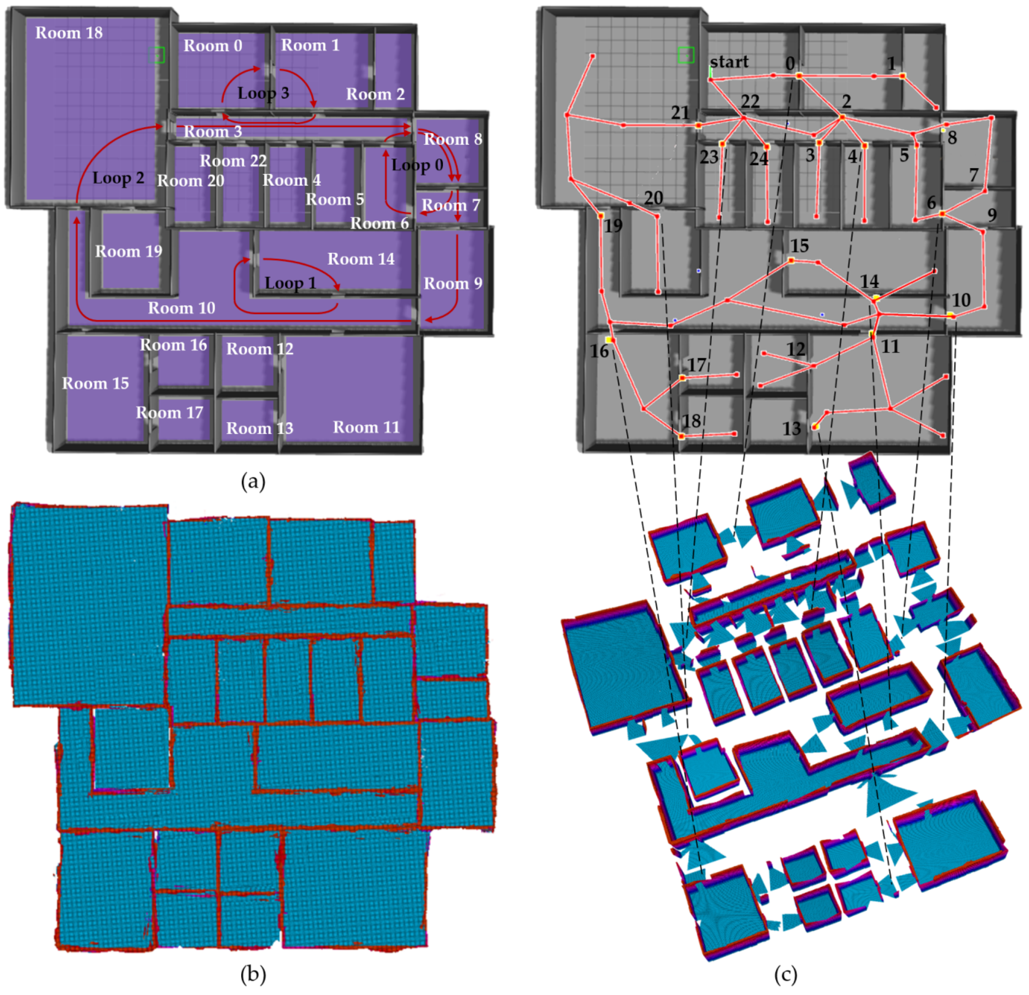


Figure 4.12 Mapping experiment in a simulated environment. The environment for the simulation is shown in (a). In (b) the resulting map when building directly the global 3D map is represented. The resulting traversability graph and the submap generated for each room is shown in (c). The detected doors are shown in the traversability graph and associated to the connection between submaps.

Comparison of 3D Submaps to Fixed Submapping of the Environment

Some works define submapping with a fixed size or a fixed distance traveled by the robot (as the work presented in [100]). In this experiment, a comparative result between submapping based on fixed traveled distance and the proposed method is shown. Experiments were conducted in the environment shown in Figure 4.12 (a) and mapping efficiency was compared using different fixed traveled distances (1.5m, 2.5m, 5m, 10m). Efficiency is considered in terms of RAM requirements and number of submaps generated as shown in Table 4.2. Regarding RAM requirements, just the approach with fixed distance 1.5m outperforms our method but generating ten times more submaps. While fixed travel distance grows, the number of submaps decreases but increasing the RAM required to store the submaps. In addition, fixed distance methods do not offer any semantic information and, while the number of submap increases, semantic interpretation is harder.

Table 4.2 Fixed traveled distance submaps VS proposed method.

Submapping method	1.5m	2.5m	5m	10m	Prop. Meth.
RAM increment [GB]	1.31	2.11	2.94	3.27	**1.55**
$n_{submaps}$	253	131	64	21	**22**

Evaluation in a Real-world Environment

An experiment in a hallway environment of University Carlos III of Madrid was performed. The area mapped corresponds to a hallway area with security doors. A submap is generated for each section of the hallway. In Figure 4.13 (a), the robot in the real environment is shown. In the back of the image, we can observe one of the security doors that the robot traverses. The resulting traversability graph and the 3D submaps built are shown in Figure 4.13 (b). The traversability graph and the 3D submaps are related through the nodes identified as doors. The robot started in a small room next to the right ending of the hallway and traveled to the right until the end of the hallway is reached. Then it plans a path through the already mapped area to continue mapping the left side of the corridor. It built 7 supernodes (resulting in 7 submaps) and 48 nodes.

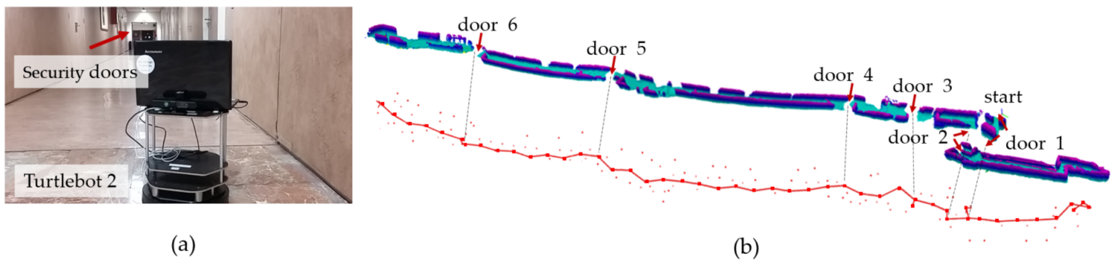


Figure 4.13 Real-world experiment for construction of the topological map, the traversability graph and 3D metric submaps. The picture in (a) shows the Turtlebot 2 robot in the real hallway environment. Resulting topological-submap representation for the university environment is shown in (b).

4.6.4 Evaluation of the Object-based Pose Graph

The construction of the object-based pose graph was evaluated in a real-world meeting room environment at University Carlos III of Madrid of approximately 56m^2. The environment was cluttered by different types of objects such as sofas, tables, chairs, lamps, plants, etc among which we only detected sofas, chairs and plants. A picture taken from the environment and the occupancy grid of the environment (built just for illustration purposes) are shown in Figure 4.14 (a) and (b), respectively. The object-

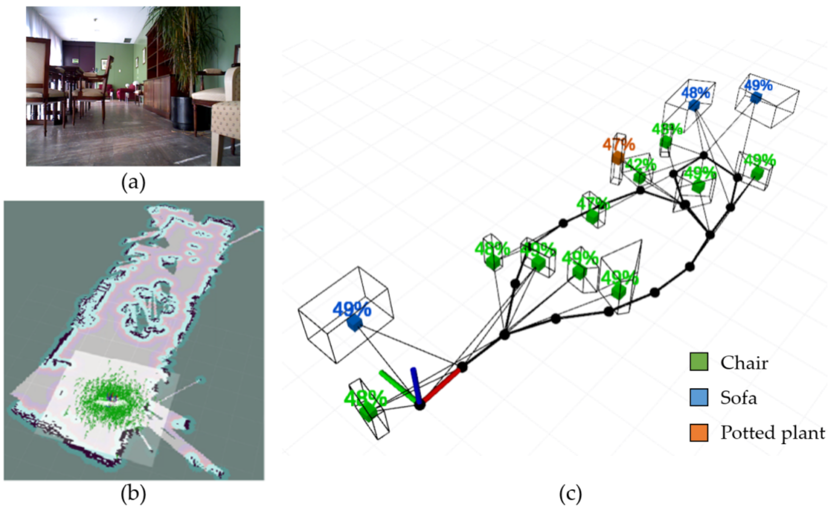


Figure 4.14 Result for object-base pose graph in a real-world environment. A picture taken from the environment is shown in (a) and the occupancy grid of the room in (b). The resulting pose graph along the color legend for the objects mapped is shown in (c).

based pose graph is shown in Figure 4.14 (c) in which the centroids of the objects are colored according to the legend. The resulting pose graph shows an occupied area in the center of the room that corresponds to the table and it is surrounded by objects detected as chairs that matches with the disposition of the environment that we see in 4.14 (a). In addition, we can find the sofas at both ends of the environment and the plant close to the table. In the occupancy grid we can find also the occupied cells that belong to the mapped objects in the object-based pose graph. The pose graph contains 17 poses and 14 objects (3 sofas, 1 potted plant and 10 chairs).

4.6.5 Evaluation of the Complete Hybrid Map

In this last mapping evaluation, we include the construction of the whole hybrid map for a real-world house environment. The house size is approximately 70m^2 and it contains 4 rooms: a kitchen, a bedroom, a living room/dinning-room and an office. The environment contains plenty of objects as it is a living space. Figure 4.15 shows five images captured by the robot in the environment, one for each room (and two for the living room).

The resulting hybrid map is shown in Figure 4.16 and it contains each of the components of the hybrid map. For illustrative purposes, in Figure 4.16 (a), a 2D occupancy grid of the environment is included where each room in the environment is labeled. In Figure 4.16 (b), we provide the topological map built from the environment. Since it is a toposemantic representation that does not contain metric information, we include it as the list of supernodes and transitions and their attributes. The mapping system identified 4 supernodes in the environment connected through 3 transitions or doors. Each of the supernodes is characterized by its identifier, its priors or connected supernodes and the nodes, objects and poses contained in the supernode. In this case, supernode 0 represents the office, supernode 1 the living room, supernode 2

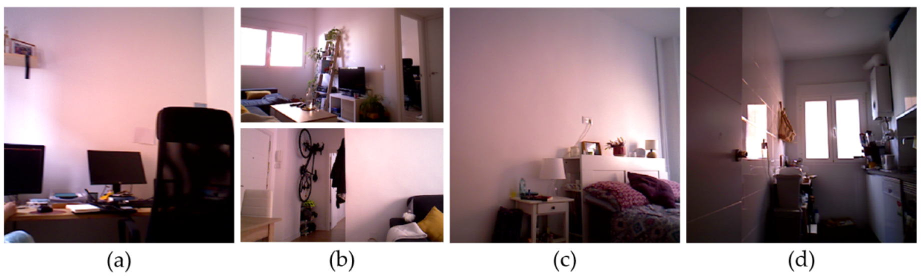
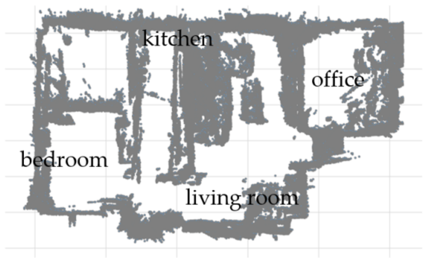


Figure 4.15 Images captured from the house environment: (a) office, (b) living room, (c) bedroom and (d) kitchen.

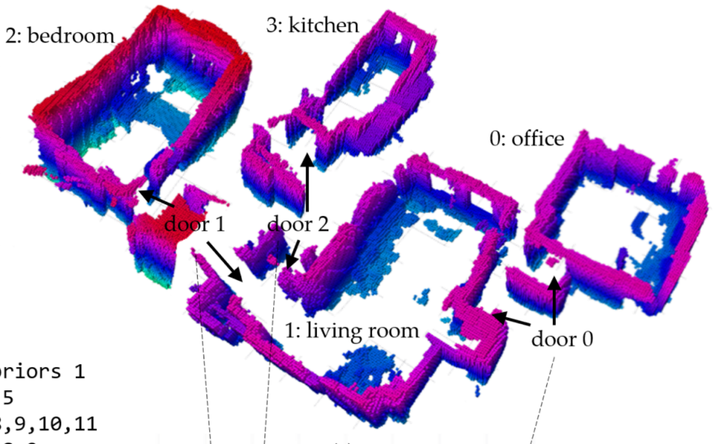
the bedroom and supernode 3 the kitchen. Observing the topological map, the room that contains more nodes, objects and poses is the living room. We can check the correspondence of nodes, objects and poses listed in the topological map through the traversability graph and the object-based pose graph (that we will describe bellow). Regarding transitions, they contain the information of the supernodes that they connect and the node of the traversability graph that is identified as the door. According to the topological map, in Figure 4.16 (c) we present the four 3D submaps of the environment, one per supernode and room. For the seek of clarity, the transitions and supernodes are indicated along the submaps. As we can see in these two representations, each transition connect the living room (s1) with one of the other supernodes: transition 0 connects to the office (s0), transition 1 connects to the bedroom (s2) and transition 3 to the kitchen (s3).

In Figure 4.16 (d), the traversability graph of the environment is shown. The traversable paths in the environment were fully mapped using 10 nodes, three of them corresponding to transit nodes or doors (1, 4 and 6) and the remaining seven to free area nodes. Node 1 represents the door that connects supernode 0 to supernode 1, so node 0 is the only node that belongs to supernode 0. Node 4 and 6 connect to the bedroom and kitchen, respectively. Thus, node 5 is the only free area node contained in the bedroom and node 7 is the only free area node contained in the kitchen. The remaining nodes (3, 2, 8 and 9) determine several traversable paths in the living room.

Finally, Figure 4.16 (e) shows the object-based pose graph of the environment and the correspondence of centroid colors to object types. In addition, each pose of the pose graph is annotated with its identifier in black and each object is annotated with its persistence probability and its identifier in red. Regarding poses, we can see the similarity between the pose graph and the traversability graph, as they are built simultaneously. Poses 0 and 1 belong to the office, 8 and 9 belong to the bedroom, 10 and 11 to the kitchen and the rest belong to the living room. Regarding objects, in this environment the system mapped a total of 23 objects: 3 chairs, 1 sofa, 4 potted plants, 1 bed, 6 screens, 1 cup, 3 bottles, 1 dinning table, 2 handbags and 1 bicycle. In the office the system mapped three screens that correspond to laptops and computer screens and two chairs. In the living room, the sofa, TV screen, two bottles of water and a cup were mapped. In addition, it mapped three potted plants, a dinning table and a chair, a bicycle and a handbag. Some errors can be discussed in the living room. In first place, detection errors such as the cup that was actually a vase and the handbag that was a bicycle helmet. These errors are caused by missdetections of the object detector used. In the second place, there were object that were not included



(a)



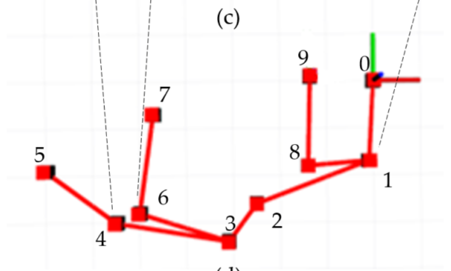
(c)

Topological map:

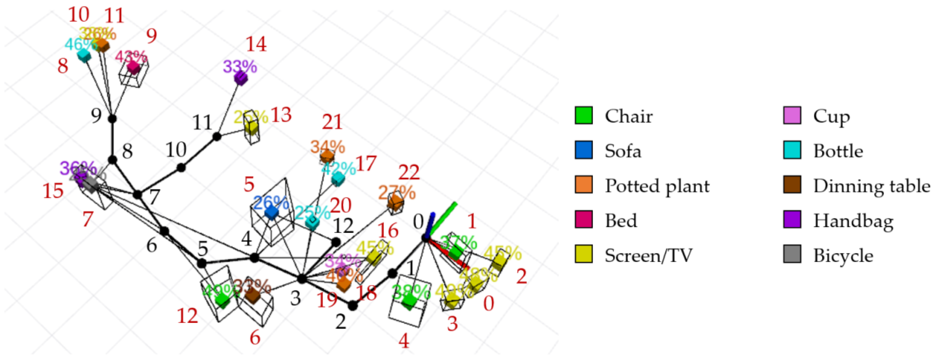
```
s0: id0 priors 1            s2: id0 priors 1
    node 0                      node 5
    obj 0,1,2,3,4               obj 8,9,10,11
    pose 0,1                    pose 8,9
s1: id0 priors 0,2,3        s3: id0 priors 1
    node 2,3,8,9                node 7
    obj 5,6,7,12,15,16,         obj 13,14
      17,18,19,20,21,22         pose 10,11
    pose 2,3,4,5,6,7,12

t0: supernodes 0,1 node 1
t1: supernodes 1,2 node 4
t2: supernodes 1,3 node 6
```

(b)



(d)



(e)

Figure 4.16 Result for the complete hybrid map in a real house environment. In (a), we provide an illustrative 2D occupancy grid of the environment just for visualization. The topological map is included in (b) as the list of supernodes and transitions. In (c), we show the 3D submaps built for each supernode. The traversability graph is shown in (d). Finally, (e) shows the object-based pose graph and the correspondence of colors for the objects.

in the map, such as a three more chairs close to the dinning table and another table between the sofa and the TV. The small dinning table and two of the chairs were not detected by the object detector, however the third chair was detected but not mapped.

In this case, it was an error of our mapping system that considered the mapped and unmapped chair as the same one. Another missdetection happened in the kitchen, where the handbag was actually detected but the screen was a detection error. In the bedroom, the bed, a bottle of water and a potted plant were successfully detected. In addition, the detected screen was actually a framed picture.

With this experiment, we showed the complete hybrid map for a real indoor environment and the connections between the different components of the hybrid map.

5

Autonomous Exploration

Maps can be autonomously built even if the movement of the robot is not autonomous and it is teleoperated or commanded by the user. In that way, the robot will acquire the information of the areas of the environment where the user drives it and the robot does not have any control over the decisions to discover and map the environment. If the objective of the robot is to completely map the environment, it is possible that, due to the gap between the commands of the user and the robot perspective, the objective is not achieved. On the contrary, if the robot moves autonomously through an exploration strategy, it can take its own decisions based on the information gathered until that moment assuring the completeness of the map. In addition, it avoids excessive user intervention and increases the autonomy of the robot.

The objective of this chapter is to develop an autonomous exploration algorithm that enables the autonomous construction of the hybrid map. Autonomous exploration and map building deals with the complexity of autonomously exploring an unknown area while acquiring the most important information to be mapped. Commonly, exploration and map-building strategies are related as the robot can build a representation of the environment while it explores (although not all of the exploration applications require mapping the environment). An exploration strategy finishes when all the environment is mapped or the goal of the application beyond the exploration strategy is reached (i. e. finding an object that has been moved or finishing a surveillance duty).

Exploration is meant to answer the question: "given the current knowledge of the environment, where should the robot move next to acquire the maximum information of the unknown environment?" The main purpose of previous research in exploration strategies is to satisfy this question using the existing technologies. One of the most common approaches for exploration is frontier-based exploration [174] which is also the foundation of most current exploration algorithms. Frontier-based exploration answers to the question presented above with a simple concept: frontiers. Frontiers are regions on the boundary between open space and unexplored space and, according to this approach, moving to the frontiers the robot will gain as much information as possible.

In this chapter, we propose a mobile robot exploration algorithm that combines frontier-based concepts with behavior-based strategies for indoor environments that will provide the robot with all the information to build the hybrid map of the environment. Differently from Yamauchi's approach [174], we propose a frontier exploration algorithm based on a cost-utility function that uses the information of the environment to select the best next frontier. Traveling through the different frontiers (until the environment is fully explored) the robot acquires the information of rooms, doors and objects along with the 3D information of the environment that will be used to generate the different components of the hybrid map.

5.1 Related Work

Mobile robot exploration has attracted researchers attention since the beginnings of mobile robot developments as it is the way to autonomously build a representation of the environment. Some of the first approaches dealt with external markers to help the exploration like the breadcrumbs in Hänsel and Gretel story. This is the case of [54], where Dudek et al. designed a robotic system that could identify, put down and pick up some markers that were used to guide the exploration. Kuipers et al. [94] presented an exploration algorithm that could built a topological map of the environment without any external help. This system recognized qualitative properties as distinctive places and travel edges between them leading to a topological representation where geometry is assimilated into local descriptions of places and edges. Other authors also gave importance to topological representations through points of interest that were identified as the robot moved [55]. Points of interest were defined as the free borders of a virtual bubble that was built around the robot occupying all the space that it perceived and growing while it moved. The exploration finished when the bubble had occupied all the environment. Points of interest can be understood as a precursor of the frontier concept presented by Yamauchi [174]. Frontiers are regions on the boundary between open space and unexplored space. Yamauchi proposed that moving to the frontiers the robot will gain as much information as possible. In this first approach to the frontier concept, the selection of the best frontier to move corresponded to the nearest frontier. Frontier-based exploration has become one of the most used and robust exploration algorithms and many authors have based their exploration algorithms in the frontier concept. In [111], authors added information about navigation and localization to the decision process for the next frontier to visit. Their algorithm maximizes the global utility which consists of information utility (maximize the amount of new information that can be acquired), navigation utility (minimize the displacement to the place) and localization utility (minimize the localization error). Other authors [4] defined an exploration strategy through multi-objective optimization of separated features where the selected candidate frontier is the one that is nearest to the individual ideal values. In the same direction, in [14] multiple features were proposed to optimize the decision making process but the selected candidate corresponded to the one that maximized a global utility function. Amigoni et al. [3] used frontier-based exploration to build geometric maps. In [2] and [87], comparisons of different exploration strategies were presented. In both works, they concluded that depending on the application the decision would vary but generally frontier-based exploration using only cost (moving to the nearest frontier) required less time. On the contrary, using cost and utility, such as the work in [70], extensive knowledge of the environment was acquired more quickly, which is important in rescue and surveillance tasks.

Meanwhile, other authors worked on improving behaviour-based approaches for exploration. In [56], a topological map called navigation chart containing the actions to move between distinctive places was presented. In [137], authors presented a behaviour-based control approach to build a topological map that established connections between rooms. A topological exploration strategy was also presented in [141] in which the behaviours were grouped in node detection, node matching and edge travelling using a Voronoi diagram.

Recently, other exploration strategies have been proposed. Such as the work presented by Fermin-Leon et al. [59] [60] where rooms are identified by topological segmentation of contours. Each room or region is associated with a node and they use Tarry maze-searching [71] algorithm to move through the environment. In [119], authors propose an exploration and rescue method based on Partially Observable

Markov Decision Processes which directly incorporates uncertainty in the decision process. In [143], a multirobot exploration algorithm in which each robot auctions for the next positions to reach is presented. Based on the observed part of the environment, the system estimates the outer border of the environment by the convex hull of the observed map and infers the structure of the unknown area. In [19], random next best views are connected through a RRT algorithm. The selection of the next best view is performed with regard to the amount of visible unmapped area and with a penalization for high costs. This work deals with 3D mapping and 2D surface inspection and shows a better performance of RRT exploration compared to frontier-based exploration for fine-grained, complex and detailed mapping. However, in house or office environments frontier-based exploration obtained a faster result.

Many recent works have also presented variations of the frontier-based exploration algorithm. In [33] and [34], authors develop a frontier exploration algorithm where the function to chose the best candidate depends on the localizability and uncertainty based on entropy. This algorithm leads to a conservative exploration strategy that maintains a good uncertainty value through loop closure and revisiting poses. Some works, such as [8] and [9], have focused on frontier-based exploration to perform navigation in an unknown environment. In these works, robots do not build a representation of the environment nor completely explore the environment, they collaboratively reach their desired goals within the environment. In [81], a method for online mapping through Gaussian Processes occupancy maps (GPOMs) is proposed. An algorithm to extract probabilistic frontiers from GPOMs is used as frontier detection. Frontier selection is performed based on information gain and path length, but just considering geometric information. They show higher performance than standard frontier-based exploration for big indoor environments. Lately, frontier exploration has also been applied to multirobot exploration establishing a routing priority for the frontiers and the robots [140]. Other works with multirobot frontier exploration have also added semantic and scene information to the decision process in order to separate the trajectories of the robots [108] or to obtain a higher reward of certain kind of areas [101]. However, these methods just use semantic information for the exploration process and build grid maps for specific areas.

In our work, an exploration strategy is presented that differentiates from the previous works in that it builds a lightweight and efficient map representation that contains geometric, topological and semantic information. This same information is taken into account to lead the exploration strategy. Semantic information is considered with regard to the traversing of transit areas and it is included in the decision process through the cost-utility function. The exploration algorithm combines frontier-based concepts with behavior-based strategies for indoor environments. The purpose of the exploration method described in this chapter is to improve the efficiency with regard to distance traveled and execution time, to increase the robustness of exploration algorithms dealing with indoor environments and to acquire the required information to build the hybrid map that includes all the geometric, topological and semantic information required for further navigation.

5.2 Frontier-based Exploration

The exploration algorithm proposed in this thesis is based on the frontier concept presented originally by Yamauchi [174]. According to that work, a frontier is a region on the boundary between free space and unknown space. Authors stated that if the robot moves to the frontiers it gains the maximum information for each movement.

In our work, the first step for the robot is to detect the frontiers and decide where in that frontier it is going to move (section 5.2.1). We propose to classify frontiers according to their semantics. As a consequence, a frontier is semantically classified as free area or transit area depending on the information obtained from the frontier detection (this information will be then stored in the nodes of the traversability graph since each node is classified as transit area node or free area node). Transit areas are defined as the frontier where the robot changes between two places (rooms) and regardless of the size of the transit area the information gain that they offer is significant. In addition, free areas are defined as the frontiers within a room that drive the robot through the different areas of that place. Semantic frontier classification is presented in section 5.2.2. This semantic information, along with the cost of moving to each frontier and the utility estimated for the frontiers, is used to determine the next best frontier (section 5.2.3). The utility estimated for each frontier depends on the size of the frontier and whether it corresponds to a transit area or not. The robot is now ready to move to the desired position. In order to do so, it executes the behaviors corresponding to the current situation, as explained in section 5.2.4. The exploration algorithm depends on the loop closure strategy explained in the previous chapter since looping situations drive the robot to an already-explored area and the exploration algorithm has to chose a new best frontier to visit to continue exploring the unknown areas. This exploration process is executed iteratively until the termination condition is reached. The termination condition in our case is defined as the absence of interesting frontiers to visit, as explained in section 5.2.5.

Once the exploration algorithm has finished, the hybrid map of the environment is completely built and can be used for further navigation. The whole process of frontier-based exploration is illustrated in the diagram presented in Figure 5.1. Sensor information from a laser, a camera and wheel odometry are used as inputs to the system to drive the exploration and, simultaneously, map the environment. When the exploration process has reached the termination condition, the hybrid map of the whole environment is available for further operations. As mentioned before, the direct output of the exploration decisions is used to build the traversability graph, this is also represented in Figure 5.1.

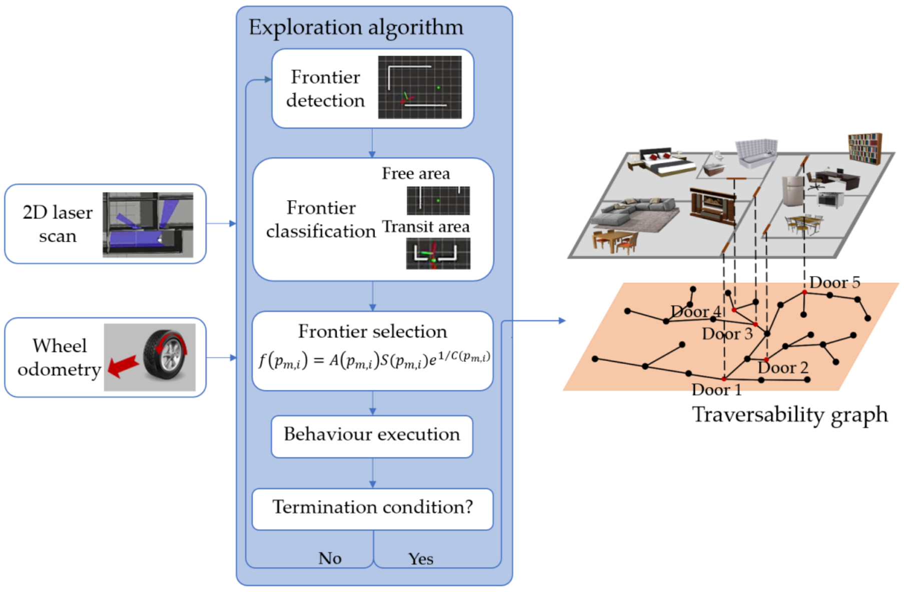


Figure 5.1 Processes involved in the exploration algorithm. The inputs of the system are the scans from a 2D laser and the odometry of the robot. The output of the system is the traversability graph of the environment.

5.2.1 Frontier Detection

Frontier detection is the process of detecting the boundaries between free-known space and unknown space. 2D laser information gives a measurement of the elements surrounding the robot. If the laser reaches an obstacle within its field of view, it will provide the distance to that object. Moreover, we can assume that all the space within that distance in that direction is free. On the contrary, if there is not any obstacle in a particular direction, the measurement in that direction will be assigned to the maximum range of the sensor and correspond to an unknown space area. Laser measurements corresponding to free, occupied and unknown space are easily separable into groups that can be associated to frontiers $P = \{\mathrm{p}_1, \mathrm{p}_2, ...\mathrm{p}_N\}$. We consider that every group of laser measurements is a frontier if:

- The value of all the measurements corresponds to the maximum range value. This means that in that direction there is not any obstacle in the seen region.
- Within the group of laser measurement there is a significant gap between the distance value of consecutive measurements. Even though range values do not reach the maximum value, it is possible to have occlusions between obstacles which are recognized through a significant difference between consecutive scans. If there is just one gap within the group, the frontier will be bounded to the gap. On the contrary, if there are two adjacent gaps (i.e. one gap with depth values from 2.3m to 4m and another gap with depth values from 4m to 2.3m) the group of measurements with the larger distance (4m, in the above example) is considered to be the gap.

This method allows to cluster laser scans into frontiers (interesting areas to discover) and already-known or occupied areas. Once a frontier, p_i, is detected from a particular robot position, it is characterized using its middle point, m_i, and the size of the frontier, where frontier and middle point are linked through their indexes. The middle point, m_i, corresponds to the position to be reached within the frontier and the size of the frontier will be used to determine the geometric utility and the semantic type of the frontier. In Figure 5.2 (a) and (b) different frontiers have been detected and their middle points are marked with a square. Both frontiers detected in Figure 5.2 (a) correspond to groups of laser measurements whose range corresponds to the maximum laser range. However, in Figure 5.2 (b), the frontier at the bottom left part of the image corresponds to a group of laser measurements that contains a significant gap between consecutive scans. After frontier detection, frontiers are classified according to the gathered information to afterwards select the next best frontier to visit.

5.2.2 Semantic Frontier Classification

Frontiers are semantically classified as free area or as transit area based on their geometric characteristics using laser and camera information. In this work, transit areas are identified with doors, $D = \{d_1^i, d_2^j, ..., d_m^k\}$, as they connect two different places. However, they could be defined differently for other environments and the autonomous exploration will work similarly. Free areas relate to frontiers within a room and drive the robot through the free area of the room until it is fully explored and the different entrances to that room are identified.

In order to classify free areas and transit areas, a preliminary door detector has been developed in this work, in which only the geometric characteristics obtained from the laser and a depth camera are taken into account. For a frontier to be considered a door, its size must correspond to a typical doorway between two coinciding segments (see Table 5.3 for parameter description). In Figure 5.2 (a) and (b), the robot identified three frontiers characterized as transit area or door (yellow) and one as free area (blue). In addition, in Figure 5.2 (b), an example of a frontier where there is a gap between measurements is shown. In this case, although the wall across the door is detected, the frontier is considered as the doorway. Once a gap is selected as a possible door thanks to laser information, camera information is used to confirm that hypothesis. A simple vertical line detector using Hough Line Transform was implemented. Vertical lines must be found close to the door frame in order to finally consider that gap as a door. In Figure 5.2 (c), the detection of the door frames of a door is shown.

Behaviors to check and confirm that a frontier detected as a transit area is really a transit area are executed in order to solve the misclassification that could occur in corners or dead-ends. As we will explain in Section 5.2.4, these behaviors are approaching to the center of the door and stopping before reaching it. From that new position the door detection algorithm is run again and it will confirm or discard that the frontier is a transit area. We assume that from this ideal position, the chances of misclassification are low and the detection is reliable.

Before continuing with the theoretical explanation of the method, we would like to include a proof of concept of the semantic frontier classification through door detection.

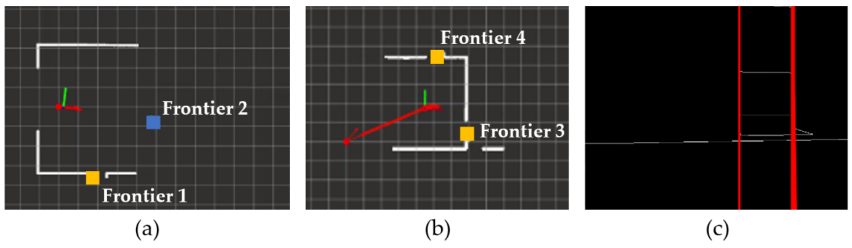


Figure 5.2 Semantic classification and different types of frontiers to be detected. (a) shows the case where a free area and a transit area are detected and (b) shows the case where two transit areas are detected, one of them is successfully detected as door even though the wall behind it is detected. In (c), door frames detection from an image using Hough Line Transform is shown.

Although detection methods are beyond the scope of this thesis, we consider that the classification results for the develop method are worth a brief description. Semantic frontier classification was tested in a simulated environment in order to evaluate its performance. The test was performed in the environment shown in Figure 4.12 (a) due to the number of doors present in the environment. Frontier classification was executed from 50 different random positions leading to the results shown in Table 5.1.

Table 5.1 Results for semantic frontier classification.

Measurement	Value	Measurement	Value
Prevalence	0.5263	Sensitivity	0.9333
Accuracy	0.9298	Specificity	0.9259
Misclassification rate	0.0702	F1-score	0.9333

The obtained results show a good performance of the classifier in the detection of doors considering the high percentages of accuracy and sensitivity. Regarding accuracy, in 92,98% of the cases the classifier detects the doors correctly. In addition, failing situations were mainly due to dead ends or corners and sharp angles to doors. After performing the behavior to check doors, most failing situation will be reduced as the doors and non-doors would be successfully identified from the new advantageous position.

This semantic classification plays an important role in the exploration algorithm as it is one of the key points for the decision process of the frontier selection. Further more, it is crucial for the map building as the construction of supernodes and 3D submaps depends directly on the successful detection of transit areas.

5.2.3 Frontier Selection through Cost–Utility Function

Newly detected frontiers and the previous ones that were not visited are grouped as $P_u = \{\mathrm{p}_{u,1}, \mathrm{p}_{u,2}, ..., \mathrm{p}_{u,N}\}$ where only the unvisited frontiers from the whole set of frontiers, P, are included. Given the set of unvisited frontiers, P_u, the robot has to decide which of them is more worthy to visit first, p_u^*. In this thesis, this decision, which depends on the expected gain for each frontier and the distance to it, is taken through a cost-utility function. A cost-utility function is defined as the function that the system tries to maximize as it represents the most optimal value. In this case, the cost-utility function, $f(\mathrm{p}_{u,i})$, for a given frontier, $\mathrm{p}_{u,i}$, results from the combination of the geometric utility, the semantic utility and the topological cost or distance. The cost-utility function must maximize the utility while minimizing the cost. These three elements are defined as follows:

- The geometric utility, $A(\mathrm{p}_{u,i})$: this utility corresponds to the size of the frontier. Its influence to the utility of the frontier comes from the fact that bigger frontiers will offer a bigger range to acquire new information of the environment. The geometric utility value ranges from 1 to the maximum number of scans that can occupy the frontier (according to the scan size), $A(\mathrm{p}_{u,i}) \in [1, 640]$.

- The semantic utility, $S(\mathrm{p}_{u,i})$: this utility gives a semantic importance to the transit areas. In spite of its small size, transit areas open to a new space that will make the robot gain valuable new information. For this reason, transit areas should have a positive influence over the utility of the frontier, $S(\mathrm{p}_{u,i}) = 30$, whereas free areas are not influenced, $S(\mathrm{p}_{u,i}) = 1$.

- The topological cost, $C(\mathrm{p}_{u,i})$: this cost refers to the topological distance that the robot will have to travel to reach the frontier. This cost is associated to the connectivity between frontiers. Consecutive frontiers will have a cost value of 1. However, if to reach one frontier the robot has to travel through already-explored frontiers (nodes of the traversability graph) its cost value will correspond to $n + 1$, where n is the number of frontiers (nodes) to cross. To summarize, the topological cost ranges from the natural values starting from 1 to the maximum distance between current frontier and non-visited frontiers, $C(\mathrm{p}_{u,i}) \in [1, n + 1]$.

Taking into account the geometric utility, the semantic utility and the topological cost, our cost-utility function for frontier selection is:

$$f(\mathrm{p}_{u,i}) = A(\mathrm{p}_{u,i})S(\mathrm{p}_{u,i})e^{1/C(\mathrm{p}_{u,i})} \quad , \tag{5.1}$$

where different utilities are multiplied whereas between utility and cost the relation is a reverse exponential. Since both utilities are bounded to values higher that 1, when they are multiplied the global utility of the frontier increases. In addition, although the geometric utility of transit areas is generally smaller than for free areas, this is compensated with the semantic utility. Regarding utility and cost, using a reverse exponential, we are penalizing the transitions that are not directly connected to the current frontier, as they imply path planning and several transitions. The impact of this penalization is evaluated in Table 5.2 since the cost-utility function, $f(\mathrm{p}_{u,i})$, is calculated for fixed geometric and semantic utility values, $A(\mathrm{p}_{u,i})$ and $S(\mathrm{p}_{u,i})$ respectively, and increasing the topological cost, $C(\mathrm{p}_{u,i})$. This results in a great penalization between cost value 1 (0%) and cost value 2 (39.35%) to avoid excessive path-planning and revisiting of nodes. As the cost increases, lower penalization increments are applied (as once the robot is performing path planning, the number of revisited nodes is not so determining). This effect is due to the reverse exponential related to the cost.

Table 5.2 Study of the relation between cost and utility and the influence of cost in the proposed cost-utility function.

$C(\mathrm{p}_{u,i})$	$A(\mathrm{p}_{u,i})$	$S(\mathrm{p}_{u,i})$	$f(\mathrm{p}_{u,i})$	% of penalization
1	60	1	**163.09**	0%
2	60	1	**98.92**	39.35%
3	60	1	**83.71**	48.68%
4	60	1	**77.04**	52.77%

The cost-utility function is calculated at each exploration iteration for each possible non-visited frontier and the best frontier p_u^* corresponds to the one that maximizes the cost-utility function:.

$$\mathrm{p}_u^* = \underset{i}{\operatorname{argmax}} \left(f(\mathrm{p}_{u,i}) \right) \ . \tag{5.2}$$

The autonomous exploration process described up to now can be summarized as the sequence presented in Figure 5.3. From a starting position the robot seeks for the best option to start exploring the environment. Two frontiers have been detected and semantically classified as free areas. Both frontiers are placed at a cost value of 1 and the geometric utility of frontier 2 is much bigger than the one of frontier 1. For this reason, in the first stage the robot decides to move to frontier 2. When the required position for that frontier is reached, the second exploration stage starts. The

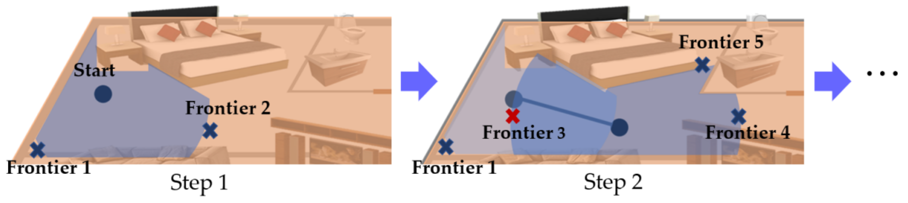


Figure 5.3 Sequence for the detection, classification and selection of frontiers in two exploration stages. Frontiers are analyzed and visited until the termination condition is met and the environment is fully explored.

robot detects three new frontiers. Frontier 3 is discarded since it corresponds to an already-visited area and frontier 4 and 5 are evaluated along with frontier 1 that was not explored in the previous stage. Frontier 1 has a cost value of 2 and the new frontiers a cost value of 1. In addition, frontier 4 opens to a much wider area, so frontier 4 will be the next one selected to explore. This sequence will continue until the termination condition is met.

5.2.4 Behavior-based Exploration Strategies

The robot can perform three different behaviors to fulfill the exploration process. The behaviors implemented are *Move to next free area*, *Approach transit area* and *Cross transit area. Move to next free area* behavior is performed for frontiers classified as free area and it reaches the middle point of the next best frontier. *Approach transit area* and *Cross transit area* are performed for frontiers classified as transit area. The robot first performs *Approach transit area* which consists on moving towards the middle point of the frontier but stopping 90 cm before reaching it (90 cm is selected as stopping distance given that it gives reliable measurement values according to the sensor used). When that approaching position has been reached, the robot checks that it is effectively a transit area. In order to check if the frontier is actually a transit area, the door detection algorithm is run again from this closer position. If the transit area has been checked, the behavior *Cross transit area* is executed. It moves the robot through the transit area and beyond until it has entered the new room. If the transit area has been discarded (from the closer position the robot has sensed it as a dead-end or corner), a new frontier to visit is sought. Each behavior requires different speed and precision conditions.

When the next best frontier is situated in a topological cost higher to 1, prior to executing the required behaviors the robot has to plan the path to reach the next best frontier. This path planning is performed using Dijkstra path planning algorithm [142] that finds the shortest path between the known nodes of the environment and executes the corresponding behaviors for each of the nodes it has to traverse.

5.2.5 Termination Condition

This frontier-based exploration algorithm finishes when the environment is fully explored or *nearly* fully explored. In the algorithm level, full exploration of the environment is equal to the absence of any frontier to visit. When the robot has explored all the frontiers that it has detected, we assume that all the reachable space by the robot has been explored. However, after some exploration time, most of the frontiers that remain unvisited belong to small regions that do not offer any additional information of the map. For this reason, we introduce the concept of interesting frontiers for those frontiers that achieve that, when all of them have been visited, the environment is *nearly* fully explored and all the relevant information has been gathered.

A frontier is considered interesting if its cost-utility value is higher than an experimentally defined value, α. If none of the remaining possible frontiers has a cost-utility value higher than the minimum considered interesting, $f(\mathrm{p}_u^*) < \alpha$, the algorithm finishes. This procedure avoids time-consuming explorations that lead the robot to areas that do not add extra information of the environment.

The minimum interesting function value has been determined experimentally in the simulated indoor environment shown in Figure 5.4 (a) in order to determine the highest value that allows *nearly* full exploration of the environment without over-exploring it. The chosen value for the minimum interesting function is the same for all the experiments shown in this chapter. In Figure 5.4, the covered area for different minimum interesting function values is shown. Figure 5.4 (b) was performed with $\alpha = [80 - 70]$; Figure 5.4 (c) with $\alpha = 60$; Figure 5.4 (d) with $\alpha = 50$; and finally, Figure 5.4 (e) corresponds to the covered area with $\alpha = [40 - 10]$ and shows a fully covered environment.

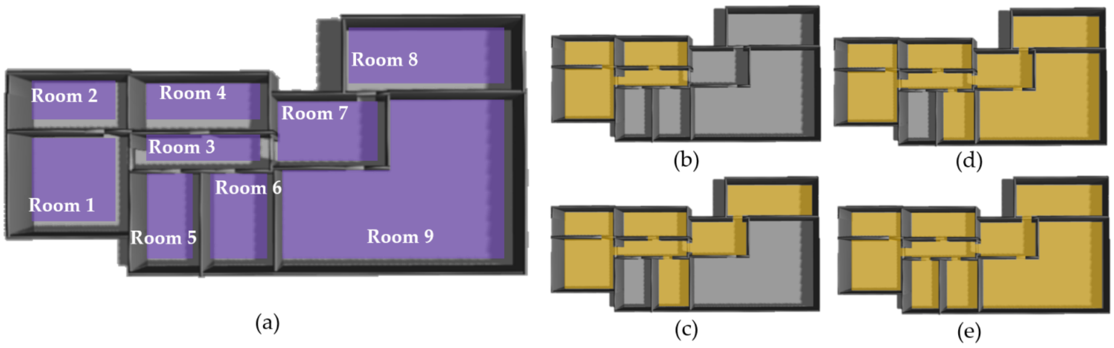


Figure 5.4 Evaluation to determine the termination condition of the frontier-based exploration. In (a) the simulated environment for the evaluation consisting of 9 rooms connected through 8 doors is shown. The subsequent figures show the covered area for different minimum interesting function values: for $\alpha = [80 - 70]$ in (b), for $\alpha = 60$ in (c), for $\alpha = 50$ in (d) and for $\alpha = [40 - 10]$ in (e).

Some of the differences can be observed at first sight, for example it is obvious that the explorations for Figure 5.4 (b), (c) and (d) are not complete, but some other differences are not so obvious without some metrics. The metrics used for determining the minimum interesting function value are the execution time, the distance traveled and the percentage of non-visited rooms. As shown in Figure 5.4 (a), this environment consists of 9 rooms and it is essential that the exploration algorithm explores each of the 9 rooms. In Figure 5.5, the execution time (minutes) is shown in red, the distance traveled (meters) is shown in orange and the percentage of non-visited rooms is shown in purple. The minimum value must guarantee that all the rooms of the environment are visited, this corresponds to α equal or inferior to 40. Analyzing the execution time and the distance traveled both minimize at 40 (within the valid values according to the number of visited rooms), so 40 will be the optimal value. However, we decided to set α to 30 penalizing the distance traveled but setting a tolerance for other situations. From now on, all the experiments took place with a minimum interesting function value for termination of the exploration of 30.

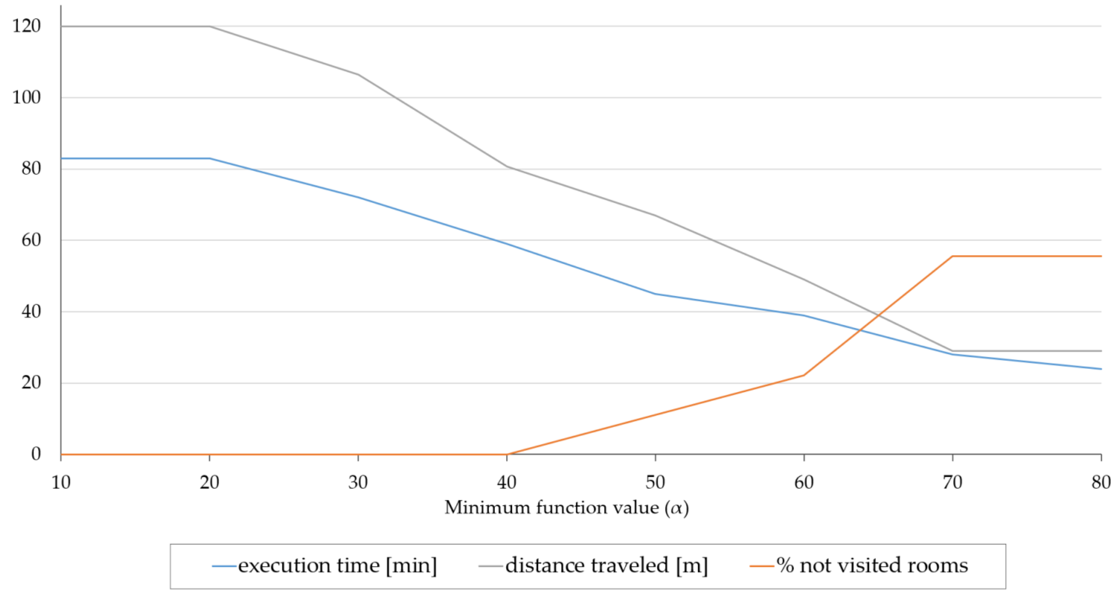


Figure 5.5 Comparison of the exploration results for the different minimum interesting values. X-axis for the function value and Y-axis for the metrics: traveled distance(m), execution time(min) and percentage of non-visited rooms.

The autonomous exploration method described covers the strategy to move the robot in order to explore efficiently the environment. Decision making was conducted through frontier-based exploration and execution was performed with behavior-based strategies. We proposed using semantic, geometric and topological information of the environment to determine the next best frontier to visit in indoor environments through a cost–utility function. The autonomous exploration process finishes when the robot has discovered *nearly* the whole environment. While the robot explores the environment, each layer of the hybrid map will be incrementally built. In the following, we include the experimental evaluation of the explained autonomous exploration method. Different simulation and real-world experiments will show the results for the autonomous exploration. In addition, comparisons to other methods are included to uphold the improvement due to the proposed method.

5.3 Experimental Evaluation

This section includes robot exploration evaluations both in simulated and real-world environments, the parameter description for all the experiments is included in the experimental setup. In addition, we include comparisons to other exploration strategies.

5.3.1 Experimental Setup

Exploration experiments were conducted firstly using Gazebo simulation environment. The performance and resulting maps were visualized using RViz since the work was developed using ROS (Robot Operating System framework) and C++. Several indoor environments were created to develop the experiments as close as possible to real situations. The simulated version of Turtlebot 2 robot is equipped with the simulated sensors Hokuyo URG-04LX laser sensor and Asus Xtion Pro camera to perceive the environment. Regarding the computational resources, for both simulated and real-world experiments, a PC with IntelCore i7-6500U CPU@2.50Hz 12GB RAM was used.

Real-world experiments in a house environment are also presented. Turtlebot 2 robot was used for the experiments and RViz visualizer was used to observe the resulting map of the environment. As in the simulated experiments, Turtlebot 2 is equipped with a Hokuyo URG-04LX laser sensor and an Asus Xtion Pro camera. The robot's speed was limited to 0.4 m/s in linear velocity and 0.7 rad/s in angular velocity.

The performance of the exploration method is going to be evaluated using the distance traveled and the time spent in the exploration. In addition, we will show the resulting paths traveled by the robot during the exploration process.

For all the exploration experiments presented in this chapter, the same set of parameters was used (maintaining the same configuration also for the compared algorithms). The employed parameters are summarized in Table 5.3.

Table 5.3 Set of parameters used for the experimental evaluation.

Parameter	Value	Parameter	Value
Maximum Laser Range [m]	5	Semantic utility value for a transit area	80
Laser ranges corresponding to front	[275, 365]	Semantic utility value for a free area	1
Laser distance to consider front free [m]	1.5	Geometric loop variance	0.9
Scan difference to accept gap [m]	0.8	Topological loop variance	0.85
Min. scans amount for interesting gap	10	Loop probability to start loop closure	0.3
Small door size (single leaf) [m]	[0.8, 1.2]	Loop probability to accept loop	0.9
Big door size (two leaves) [m]	[1.6, 2.4]	Number of iterations to discard loop	5
Safe distance to approach door [m]	0.9	Frontier reaching tolerance [m, rad]	0.1, 0.02
Distance to cross door (after door)	0.5 m	Min. function value for termination	30

5.3.2 Evaluation in Simulated Environments

Exploration was tested in the simulated environments shown in the map building experimental evaluation (Figures 4.7) that contains two small non-cyclic house-like environments and a small cyclic environment. In this experimental evaluation, we are showing the path traveled by the robot during the exploration of each environment, Figure 5.6. In addition, we include average execution time and distance traveled for the exploration of each environment in Table 5.4.

The paths traveled by the robot during the exploration process are shown in Figure 5.6. Squares are used to represent the detected frontiers, in red free area frontier and in yellow transit area frontiers. The starting and ending position for each exploration is also indicated. As shown, the whole environment was explored in the three cases and all the doors were successfully detected as transit areas. The sections of the paths that the robot drove multiple times due to path planning or loop closing are also observable.

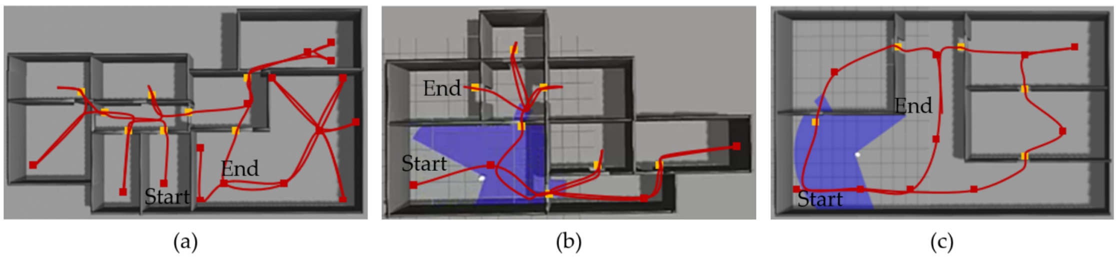
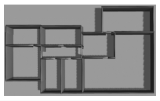
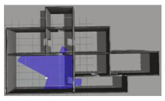
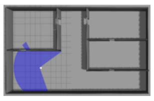


Figure 5.6 Paths traveled during the exploration process in several simulated environments. The squares in red indicate the free area frontiers and the the squares in yellow the transit area frontiers.

Execution time and distance traveled during the exploration process in the three environments using the proposed method is shown in Table 5.4. In the next experimental evaluation, we include a comparison to other exploration methods for the second environment.

Table 5.4 Average time and distance traveled during exploration in the simulated environments.

Environment			
Execution time [min]	59	44	52.2
Traveled distance [m]	84.46	57.5	99.43

5.3.3 Comparison with Other Exploration Strategies

To validate the results obtained for non-cyclic indoor environments, a comparison to other state-of-the-art algorithms is presented. The proposed algorithm was compared to: wall-following exploration [89], frontier-based exploration using cost function [174] and frontier-based exploration using GB-L cost-utility function [70]. Wall-following algorithm consists in simply following the walls until the starting point is reached again. This algorithm is not valid for cyclic environments given that it would explore just one of the loops. Regarding frontier-based exploration using a cost function, we used the same structure as for the proposed algorithm but the cost function is adapted to meet the requirements of Yamauchi's proposal [174]:

$$f(\mathrm{p}) = \lambda / L(\mathrm{p})) \quad , \tag{5.3}$$

where $L(\mathrm{p})$ refers to the topological length to reach the frontier and λ is a constant.

Finally, frontier-based exploration using GB-L cost-utility function works with the cost-utility function included in eq. (5.4) presented by Gonzalez et al. It is a well-known and robust definition for frontier-based exploration where $A(\mathrm{p})$ refers to an estimate of the unexplored area visible from p, $L(\mathrm{p})$ refers to length or cost of the path (as only topological distance is considered in this work, $L(\mathrm{p})$ refers to connectivity and not geometrical length) and λ weighs the new information and the cost.

$$f(\mathrm{p}) = A(\mathrm{p}) e^{-\lambda L(\mathrm{p})} \quad . \tag{5.4}$$

The four exploration strategies were implemented and their performances were compared according to average execution time and average distance traveled for the middle-size environment shown in Figure 5.6 (second column). Execution time and traveled distance were computed for 10 exploration sessions for each algorithm from random initial positions. Average results are shown in Table 5.5. All the exploration strategies were evaluated under the same conditions (removing non-interesting frontiers and same parameter definition).

Table 5.5 Comparison of different exploration strategies according to execution time and traveled distance: wall-following, frontier exploration with cost function, frontier exploration with GB-L function and the proposed method.

Method	Execution Time [min]	Traveled Distance [m]
wall-following	56	97.2
cost function	51	57
GB-L function	43	59.4
proposed algorithm	**44**	**57.5**

Frontier-based exploration algorithms had a much better performance than wall-following, which could be predictable. The experiment that led to a lower execution time was GB-L function frontier exploration and the experiment that traveled a lower distance was the cost function frontier exploration. However, the proposed algorithm gave the overall best performance considering both average execution time and average traveled distance.

5.3.4 Evaluation in a Real-world Environment

Exploration experiments were also performed in the three real-world environments presented for mapping. Two houses shown in Figure 4.10 and Figure 4.16 and a corridor area of University Carlos III of Madrid shown in Figure 4.13. The traversability graphs and the traveled paths for the house environments are shown in Figure 5.7 and Figure 5.8. It is observable how the nodes of the traversability graph correspond to the frontiers of the exploration process. Frontiers in red represent free areas and frontiers in yellow represent doors. In addition, the edges of the traversability graph correspond to the paths traveled by the robot. The first house environment was totally explored in 12.2 min and the robot traveled 34.5 m and for the second one the exploration time and distance were 5.1 min and 27.26 m, respectively. For the exploration shown in Figure 5.7, the robot started in a room (left) and crossed its door to a corridor connecting to the other rooms. It enters in the next room (right, down) and explores it until it is fully covered and has to plan a path to exit the room and continue exploring the corridor. Going up through the corridor two other rooms are found. The first one (right, up) is explored and then a path is planned to exit and explore the last room. When that room is fully covered the exploration finishes. For the exploration shown in Figure 5.8, the robot starts in a room (right) and exits through its door to another room. From there, the system has to decide to go up or to the left to continue the exploration. It continues to the left until two rooms are found. Firstly, the system explores the room in the left, when it is fully explored, it plans a path to exit the room and visits the other room. When that other room is fully explored the system plans a path to explore the area of the second room that was still not explored and, then, the exploration finishes.

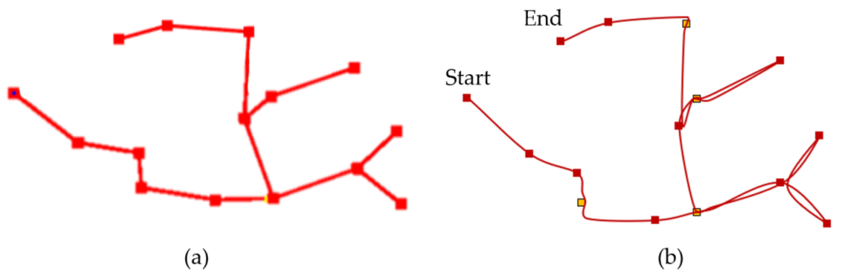


Figure 5.7 Real-world experiment for exploring a house environment: (a) shows the resulting traversability graph for the house (red squares represent nodes); (b) shows, the path traveled by the robot during the exploration (red squares are free area frontiers and yellow squares transit area frontiers).

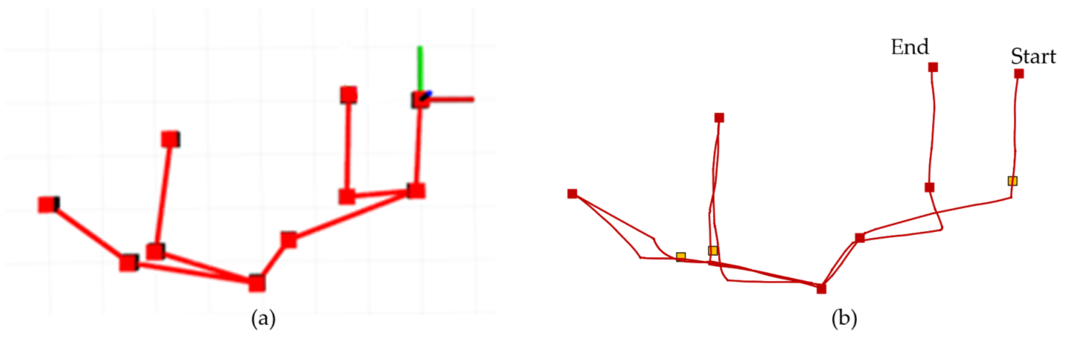


Figure 5.8 Real-world experiment for exploring a house environment. Similarly to Figure 5.7, the path traveled by the robot the traversability graph, free area frontiers and transit frontiers are shown.

The traversability graph and the traveled path for the corridor environment is shown in Figure 5.9. It is observable how the traversability graph corresponds to the path traveled by the robot. Frontiers are not included along the path as in this environment 48 frontiers were detected. The environment was totally explored in 19.5 min and the robot traveled 178 m. The robot starts in a small room next to the corridor area. When it reaches the corridor area, it decides to explore it to the right first. When the right end of the corridor area is reached, it plans a path to continue exploring the corridor to the left until the exploration is ended.

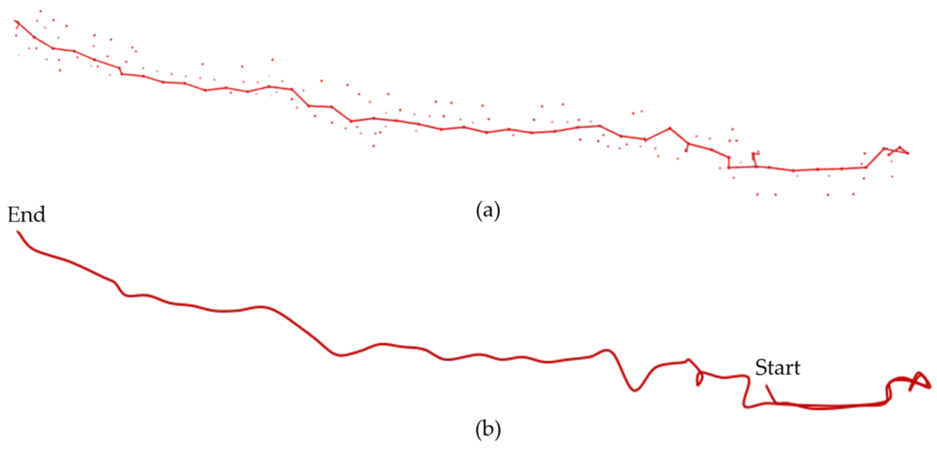


Figure 5.9 Real-world experiment for exploring a corridor environment. In (a), the resulting traversability graph for the house is shown. In (b), the path traveled by the robot during the exploration.

The presented exploration method along with the building methods described in Chapter 4 allows to autonomously build an initial representation of the hybrid map that contains information of all the components and the links between them. As shown the method outperforms other state-of-the-art frontier-based exploration methods and completes a successful exploration, both, in simulated an real-world environments.

6

Map Adaptation for Non-static Environments

Relying on static representations of the environment limits the use of mapping methods in most real-world tasks as real-world environments are not static and continuously undergo changes. These changes present a major challenge for mobile robots given that maps rapidly become outdated. An outdated map can cause problems in mobile robot operation such as crashing with unmapped elements, wrong elections during path planning, localization failures or impossibility to perform higher level tasks such as looking for objects. This is a problem that needs to be solved in order to enable mobile robots to autonomously operate in real-world environments.

The changes that affect an environment can be classified as high-dynamic changes (or high dynamics) and low-dynamic changes (or low dynamics) [25, 125, 163]. High dynamics refer to changes that occur while the robot is present in the environment (i.e. a person walking or a car driving through a road). Dealing with these changes consists in registering the dynamic elements and track them in order not to collide with them. In contrast, low dynamics refer to changes that happen while the robot is elsewhere or not sensing (i.e. returning to a room after any rearrangement of the furniture or daily register of cars in a car park). These changes are harder to detect since the robot has to compare its current representation with the new state of the environment and this can be as demanding as mapping again the whole environment. However, map adaptation consists in determining the strategies to efficiently acquire the new information without the need to remap again the environment. In this thesis, given that we are mainly dealing with mapping, we are only going to handle low dynamics because they affect the structure of the map. Managing the high dynamics is important for moving safely in the environment, but high-dynamic elements are rarely worthy to get mapped.

Given an initial representation of the environment, map adaptation deals with modifying the original representation so it matches with the current state of the world at every mapping session. Map adaptation implies change detection and map management. Change detection approaches based on vision mainly use features or objects as perception elements and compare the perceived ones with the expected ones in order to determine the changes. Map management refers to the procedures to include the change that has been detected. Some approaches are just interested in the static parts of the environment, so whenever a change is detected that element is removed from the map.

Regarding our hybrid map, map adaptation could concern every of the components of the map although it will be more useful for some components than others. The topological map represents the structure of the environment with respect to rooms and doors. Any change in the structure will imply a reform in the building, which does not happen very often, and it will involve the change of the other components of the hybrid map. For these reasons, we think that whenever it is necessary to include any change in the topological map, it is better to remap the whole environment. However, we approach small temporary changes, such as closed doors. The traversability graph represents the skeleton of the free areas in the environment where the robot can move. This representation is highly affected by the changes in the environment since paths can become impassable or new paths can appear due to moved elements. Similarly, the object-based pose graph is highly affected by the changes as, in addition to including the possible paths in the environment, it contains information of objects and their position. This requires to represent any object that has been moved even if it does not affect the passable areas (i.e. a book that has been moved from a bookshelf to a table). Finally, the 3D submaps are the most affected by the changes in the environment as they contain the exact voxel-wise distribution of the environment and the most slight change (in addition to all the aforementioned changes) could affect the representation.

In the current state of our work, we have only dealt with two types of changes: paths that have become impassable, that are detected and managed like explained in Chapter 3, and changes in object positions in the object-based pose graph, that is going to be described in the following. We acknowledge that there is still work to do in making the whole hybrid map robust against low-dynamic changes, but managing changes in the transitability of the environment and adapting the 3D submaps to object and appearance changes remains as a future work.

In this chapter, we are performing map adaptation in the object-based pose graph. We are detecting whether an object remains static in the environment or whether it is added, moved or removed from the environment. In order to handle these possible changes, we present a probabilistic method in which each detected object will be assigned a probability to remain static during the subsequent mapping sessions or, on the contrary, be movable. For this purpose, we will use some attributes of objects (presented in Section 3.4): the persistence of an object, $p(o_i)$, and their active parameter a_i. We are going to use the persistence of an object to determine how static the object is. A higher persistence value means that the object is unlikely to be moved. We will use the active parameter to build the active map of the environment, that indicates which of the objects were present in the last mapping session. This parameter is important because we are not removing any of the detected objects, so if a chair is moved from one position to another, we will have that chair represented twice. In that case, the first representation will become inactive while the second one will remain active as long as the chair is in that position. If the chair remained in the second position for a long time, the persistence of the chair in the first position would progressively decrease and the persistence in the second position would progressively grow.

In the rest of this chapter, we will first present the related work in vision-based map adaptation to low dynamics in order to highlight the novelty of the proposed method given that it deals with objects for change detection and quantifies the movability of an object (opposite to binary approaches). We will, then, present our method for pose graph adaptation over time and how it can benefit the semantic understanding of the environment by inferring the movability of object classes. We will finish this chapter with an experimental evaluation that validates our approach and a discussion.

Our main contribution in this chapter is the design of a novel method to maintain object-based pose graphs in low-dynamic environments based on a probabilistic function that captures how static or movable an object is according to the experience of the robot.

6.1 Related Work

Methods that adapt to the changes occurring in real-world dynamic environments have received much attention from the robotics community in the last decade. Most low-dynamics approaches are based on features extracted from laser scans, images or point cloud information [10, 31, 114]. However, such developments have not occurred equivalently with other sources of information such as objects.

Regarding feature-based methods, some authors propose a binary classification of these features just considering if they are stable or not. In [163], a pose graph is updated to remove the scans that no longer match the environment. In such a way, the resulting map is built with the scans that belong to static objects. Similarly, in [68], a grid map, initialized with the architectonic map, is augmented with the features that persist in time.

Other authors found binary classification very limited and defined methods to measure and quantify the stability of the features. The Feature Stability Histogram (FSH) was proposed in [10], where image features are gathered for each node of a topological map. Over time, a voting scheme is used to register the local feature stability and the resulting map is built with the most stable features. With the same spirit, the work presented by Bürki et al. [31] applies a ranking function that estimates how likely a landmark is observable under the current situation. Top-ranked landmarks are stored for the resulting map. In [114], Meyer-Delius et al. present a method for grid maps in which the belief about the occupancy of a cell is represented with hidden Markov models. The resulting map includes the change probability for each cell, which is a novelty in contrast to the aforementioned works. Rosen et al. [132] proposed the persistence filter, which provides a probabilistic Bayesian belief over the persistence of features in a semi-static environment. In [10, 68, 163], mapping is only carried out for the most stable features, which is a limitation since the information that could be inferred from the dynamic objects is overlooked. In [45], another solution is proposed based on maintaining different representations following a memory model. The sensory memory stores the most current features. An attention mechanism selects which features are moved to the short-term memory. And finally, through rehearsal, static features can be committed to the long-term memory.

Other works focus on maintaining the most updated version of the environment by having a record of the static and current dynamic elements. In [99], a pose graph based on point clouds uses matching techniques to accumulate the aligned data and remove the outdated ones. A more sophisticated approach is introduced in [25], where a pose graph is maintained using the belief of scan matches given a certain robot position and observations. A pose is removed if its belief drops below a tolerance value. These approaches, although partially representing the environment, neglect prior situations and forget about the former presence of elements and their locations. Some works solve such issues by maintaining multiple representations. The work presented in [43] keeps maps from different experiences that are evaluated simultaneously selecting the most adequate one for the current situation or creating a new one. Similarly, in [18], several representations are maintained simultaneously from multiple timescales, allowing the robot to detect patterns. Patterns between different experiences are also sought in [91, 92] through spectral representations that model the frequency of appearance of different features.

The main drawback of the aforementioned works is the lack of semantic meaning of the information stored in the map. Features extracted from images or point clouds and scans are abstract sources for which the correspondence between the point and the semantic element it belongs to is not straightforward. For this reason, some authors started to focus on objects as the elements to map the environment. Relevant works based on objects in static environments are [113, 134], where robust pose graphs of indoor environments and object reconstruction are proposed. Recently, an adaptation of this work for coping with high-dynamic environments was proposed in [171]. Xu et al. use semantic, geometric and motion information for object tracking and pose estimation within the pose graph. Other solutions for high-dynamic environments have been proposed [17, 103, 133, 175]. In [103], a static weighing method estimates whether an object is static or not based on the Euclidean distance between object edges in two situations.

Regarding low-dynamic environments, changes at an object level can be detected inferring if they are static or movable. In [58], Fehr et al. introduce an approach where a map of the static environment and a database of the discovered objects is maintained over time. Both the map and the objects are 3D reconstructions that are refined as the robot discovers the environment. In [112], several representations of the environment are maintained and overlaps between objects and representations are checked for changes. An object-based pose graph is developed in [167], where the most up to date representation of the environment is maintained by merging new objects and removing old ones. In [26], Bore et al. propose tracking of objects in 3D maps while mantaining an updated representation of the environment over time. When an object has disappeared from its mapped position, the system looks for it until its new location is found.

In contrast to the related work, we propose to maintain the object-based pose graph of our hybrid map over time and capture the movability of the objects. To the best of our knowledge, such works have been proposed for features but not for objects. In addition, a new definition for describing the probability of an object to be in an already-mapped position is presented. Our resulting map is a probabilistic object-based pose graph where static and movable objects are included and improved object classification is obtained.

6.2 Pose Graph Adaptation over Time

The initial pose graph captures the objects and trajectory while the robot explores the environment for the first time. This process implies the generation of robot poses, mapping the detected objects and connecting poses with objects. Once the first mapping session has finished, every time the robot visits the environment again, the pose graph has to be updated. For every mapping session, m, the robot is assumed to follow the same path as in the first mapping session (as we said before, we are not updating the traversability graph nor the paths up to now) although it can drive it partially or in different directions. Updating the pose graph implies adding new objects and detecting whether the already-mapped objects are still present or have been moved/removed. While objects detected within one mapping session are updated and merged online, matching objects between different mapping sessions is performed offline when the mapping session has finished. Therefore, object probability is just evaluated and modified once and efficiency is improved given that unnecessary evaluations are avoided while the robot is moving.

In the initial pose graph, objects are initialized with the default value for the persistence of the object and active parameter. As the persistence of the object works like a probability that ranges between 0 and 1, the default initial value is 0.5 times the confidence of the detection system. This value will be close to 0.5 since most objects are detected with a high detection confidence. This is a good initial value given that we still do not know if the object will tend to be static or movable (this value could also be initialized differently according to the semantic class of the object). In addition, all the objects are labeled as active in the first mapping session. Managing the low dynamics of the map consists in updating the already-mapped objects while the robot visits the environment, especially varying their persistence value and active parameter, and adding the newly detected objects. The process of building the initial pose graph and adapting the pose graph to the changes of the environment is illustrated in Figure 6.1.

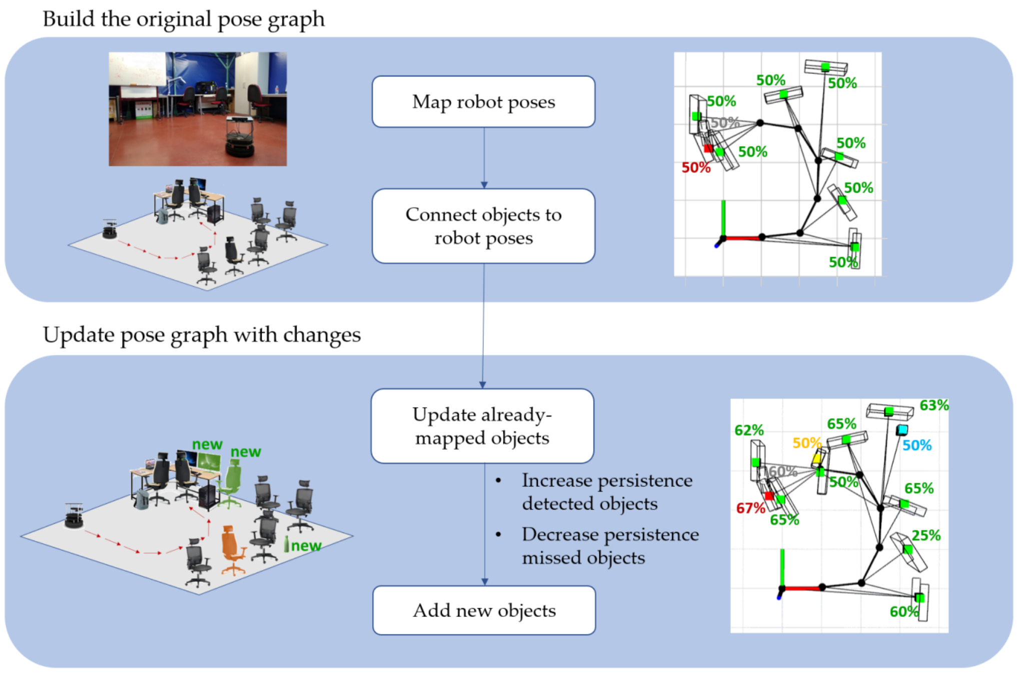

Build the original pose graph
Map robot poses
Connect objects to robot poses
Update pose graph with changes
Update already-mapped objects
Increase persistence detected objects
Decrease persistence missed objects
Add new objects
new
new
new

Figure 6.1 Illustrative example of building the initial pose graph and the adaptation of the pose graph. In the environment (left), one chair, a screen and a bottle were newly added in the second mapping session and one chair was removed. In the pose graph (right), the new objects are included with the initial probability (0.5), the object that was removed decreases its persistence probability and the rest of objects increase it.

An illustrative example is presented to represent the effect that changes, such as moved objects and added objects, has in the map, especially in the persistence value of the objects. In the following, we will describe the mechanisms to perform both tasks and also how the system can infer object class movability based on single objects movability.

6.2.1 Updating Already-mapped Objects

Map adaptation requires to identify and register the changes that affect already-mapped objects. In a new mapping session, an already-mapped object can remain in its place or be missing from its previous place. In addition, since the robot does not need to execute the same exact path as in the previous mapping session (it can travel the path partially or in opposite direction), it is possible that an already-mapped object is not visited. For that reason, in order to manage the map over time, we consider different lists of objects: already-mapped objects, expected objects and observed objects. As we have been already referring to during this chapter, the list of already-mapped objects contains all the objects that have been detected in the previous mapping sessions. An object of the already-mapped object list becomes an expected object when it was already mapped from the current robot pose and its mapped position enters the frustum of the camera, $c_i \in F$, where c_i refers to the centroid of the object and F denotes the frustum defined by the vertical and horizontal angles along with the minimum and maximum detection distances of the camera. Finally, the list of observed objects contains all the objects that have been perceived during the current mapping session.

As the robot moves, the observed objects are registered. When the mapping session has finished, the register of expected objects is compared to the observed objects detected during the session. In this process, several situations can appear: an expected object was observed in the current mapping session, an expected object was not observed or the observed object was not expected and seems to be new. In this section we will tackle the first two situations, while the addition of new objects will be presented in the next section.

If an expected object has been observed again, it means that the object has remained or gone back to an already-mapped position and it will be more likely to find that object in that position in the future. This implies that the object has to be marked as active and its persistence probability should be increased. The persistence probability of an object is increased when it has been observed again according to:

$$p(o_i)_m = \frac{s(c_i^m, c_i^{m-1})p(o_i|I_k) + \xi + p(o_i)_{m-1}}{2} \quad , \tag{6.1}$$

where $s(c_i^m, c_i^{m-1})$ refers to the similarity between the centroid position of both detections according to their Euclidean distance, $p(o_i|I_k)$ is the detection confidence of the object detector, ξ is a marginal value that relates to the false-negative rate of the detector and $p(o_i)_{m-1}$ and $p(o_i)_m$ refers to the previous and current persistence value for the object, respectively. The similarity takes values between 0 and 1, where the higher the measure is, the closer the two detections are; and it is computed as follows:

$$s(c_i^m, c_i^{m-1}) = \frac{1}{1 + \|c_i^m - c_i^{m-1}\|_2} \quad . \tag{6.2}$$

If an expected object is not detected again, it probably means that the object has been moved to other place or removed from the environment. This implies that the object has to be marked as inactive and the probability of finding it in that position on the future should be reduced. In this case, there is not any detection and thus the probability of detecting that object, $p(o_i|I_k)$, is zero and the new persistence probability is decreased by simplifying eq. 6.1 as follows:

$$p(o_i)_m = \frac{\xi + p(o_i)_{m-1}}{2} \quad . \tag{6.3}$$

Given the differences in the path between the mapping sessions, it can also occur that an already-mapped object is not included in the list of expected objects. This means that the place where that object is has not been visited by the robot. In our map adaptation method, the objects that were not visited are labeled as unknown in their active parameter and the persistence value remains unchanged since it can not be affected by the current mapping session.

6.2.2 Adding New Objects

When adapting the map over time, we consider that new objects can appear at new places. The objects to add are those ones detected by the robot that do not correspond to any of the expected objects. Before adding a new object, we compare the new object also with the whole list of already-mapped objects to verify that the object was not seen before from any other pose. If it was already mapped, a new connection is created that links the current pose with the already-mapped object. Otherwise, the new object is added and annotated with its persistence probability, class and cuboid. Given that the object was detected in the current mapping session it will be also marked as active. The persistence probability for newly added objects is calculated similarly to the initial probability (first mapping session), as described in Section 4.2, according to which it is assigned an average value since we do not know still if the object will tend to be static or movable:

$$p(o_i)_0 = 0.5p(o_i|I_k) \quad . \tag{6.4}$$

Updating the already-mapped objects and adding new objects are the two strategies that we have designed to adapt an object-based pose graph over time. This strategies are based on determining the active objects in each mapping session and learning about the movability of single objects while the robot visits the environment.

6.2.3 Inferring Object Class Movability

The knowledge that the system acquires from single object movability can be used to learn higher-level behaviors of the environment. Given that the system knows how movable or static the different individual objects are, that information can be abstracted to object classes. In this sense, if the robot observes multiple cups in changing positions, all the cups will have low persistence values and it could infer that the object class cup is highly movable. This achievement, could seem naive since it might be common sense and could be provided to the robot as prior knowledge. However, it might not be so trivial in different environments and the prior knowledge could not fit every situation while learning the information from the environment will always succeed. For example, given common sense, we could say that cups are highly movable. This is the case if the robot operates in cafeteria and cups appear in different positions. On the contrary, if the robot operates in a household shop, cups are likely to be static on the shelves and common sense will fail. Our method will succeed as, in the first case, cups will have a low persistence value and the object class cup will be identified as highly movable, and in the second case, cups will have a high persistence value and the object class cup will be identified as mostly static.

In order to infer object class movability, we group the individual object probabilities according to object class after each mapping session. This reveals valuable information about object movability and allows to classify static and movable objects. Movability, $M_{m,a} \in [0, 1]$, is the measure that captures whether an object class, a, tends to be static or movable given the information gathered until a mapping session m. Movability is the complement of the mean object probability for all the objects that belong to an object class, n_a, defined by:

$$M_{m,a} = 1 - \frac{\sum_{i \in a} p(o_i)_m}{n_a} \quad . \tag{6.5}$$

Movability for different object classes is inferred by the robot based on its own experience in the environment.

6.3 Experimental Evaluation

Real-world experiments are presented for map adaptation obtained with a mobile robot traveling through an environment during a month. In the experimental evaluation, we first analyze two mapping sessions and we see in detail how the persistence probability evolves. Later, we evaluate all the mapping sessions showing the results for persistence in the individual-object level and object-class level after the month of operating in the environment. Then, we compare our system to binary classification and we include a discussion where we compare our system qualitatively and quantitatively to other approaches.

6.3.1 Experimental Setup

Experiments were conducted in an indoor environment (15 x 6) m^2 using a Turtlebot 2 robot equipped with an Asus Xtion Pro depth camera. All the processing took place on a PC with IntelCore i7-6500U CPU@2.50GHz 12GB RAM. The robot gathered information in 20 different mapping sessions during a month where 22 object were present in the environment: 12 chairs, 3 sofas, 2 cups, 3 bottles, 1 plant and 1 laptop. Images gathered by the robot during some of the mapping sessions are shown in Figure 6.2. Red boxes in mapping sessions 1 and 2 highlight the changes in the environment with regard to mapping session 0.

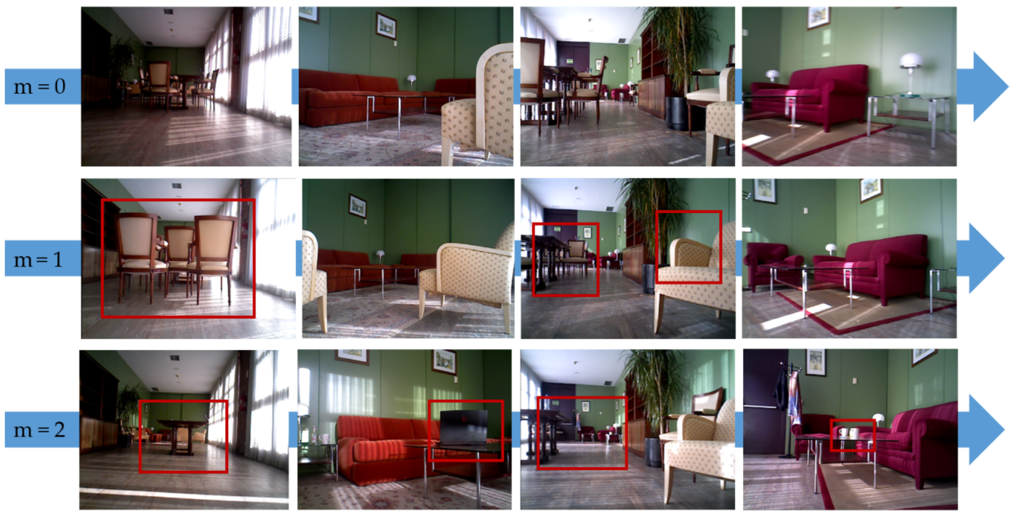


Figure 6.2 Sample of images captured by the robot in three different mapping sessions. Changes as movement of chairs, presence of new objects such as cups or bottles are introduced between mapping sessions (red boxes).

Regarding parameter description, in this experimental evaluation, the false-negative rate, ξ is assumed to be 0.

6.3.2 Two-mapping-session Evaluation in a Non-static Environment

This first experiment evaluates the performance of the mapping system and map adaptation to the changes in the environment after a second mapping session. For this purpose, two mapping sessions ($m = 0$ and $m = 1$) are evaluated in detail. The initial map, generated in mapping session 0, is shown in Figure 6.3 (a) where the traversable path in the environment is created and connected to the objects detected in this first mapping session. Figure 6.3 (b) shows the active elements for the second mapping session which refers to the objects that were detected in that session. Finally, Figure 6.3 (c) shows the resulting map after the two mapping sessions where all the objects detected during the two mapping sessions are included. Every object is represented using its cuboid, object class (color-coded) and the connections to the poses where they were detected. In addition, we include the persistence probability of each object on top of its centroid. These persistence probabilities are also included in Table 6.1 to ease the reading.

Table 6.1 shows the detail of object probabilities. In the first mapping session, $m = 0$, 14 objects were detected. Probability values closer to 0.5 belong to objects that were reliably detected by the object detector and as the values decrease the more uncertain the detector was. In the second mapping session, $m = 1$, 10 objects were detected, 6 of them in a new position different from $m = 0$. The objects that were not detected in the second mapping session are given a persistence value of 0 and the objects that did not enter the frustum of the camera are listed with "-", since the

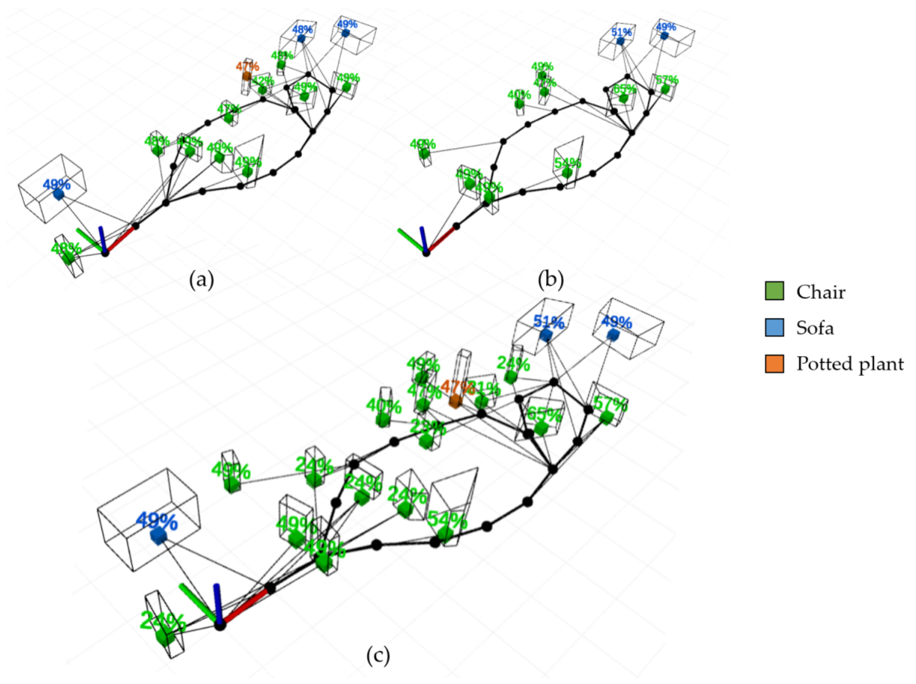


Figure 6.3 Result for two mapping sessions. In (a) the objects of the first mapping session are included (10 chairs, a plant and 3 sofas). In (b), the objects detected in the second mapping session are shown (9 chairs, some of them were in a different position than in the previous mapping session and 2 sofas). Finally, in (c), the adaptation of the map to the new situation is shown.

Table 6.1 Object probability for the two first mapping sessions.

Object	Type	m = 0	m = 1	m = 0 & m = 1
0	Chair	0.4919	0.00	0.2460
1	Chair	0.4954	0.00	0.2477
2	Chair	0.4901	0.5956	0.5428
3	Chair	0.4961	0.6477	0.5719
4	Chair	0.4981	0.8087	0.6534
5	Sofa	0.4861	0.5414	0.5138
6	Sofa	0.4961	-	0.4961
7	Chair	0.4896	0.00	0.2448
8	Chair	0.4279	0.00	0.2139
9	Chair	0.4733	0.00	0.2366
10	Plant	0.4785	-	0.4785
11	Chair	0.4856	0.00	0.2428
12	Chair	0.4864	0.00	0.2432
13	Sofa	0.4904	-	0.4904
14	Chair		0.4975	0.4975
15	Chair		0.4978	0.4978
16	Chair		0.4787	0.4787
17	Chair		0.4983	0.4983
18	Chair		0.4030	0.4030
19	Chair		0.4980	0.4980

system does not know if those objects were present in the second mapping session. When combining the two mapping sessions, $m = 0$ & $m = 1$, for the objects that were not detected in the second mapping session (inactive), the system decreases their probabilities. For the objects that were detected in both mapping sessions, the system increases their probability and finally, for the objects that were not visited in the second mapping session the probability remains.

6.3.3 Long-term Inference of Object Movability

After validating the performance of the proposed method for two mapping sessions, we want to validate the map adaptation after visiting 20 times the environment. The resulting map from each mapping session is used as the initial map for the next mapping session and each map contains all the objects detected until that moment along with their presistence probabilities and active parameter. The initial map (first mapping session) and the resulting map after 20 mapping sessions are shown in Figure 6.4 (a) and (b), respectively. The active map for the last mapping session (objects present in that mapping session) is shown in Figure 6.4 (c). In that mapping session, the robot detected 3 sofas, 3 chairs and a plant. In addition, if we define that a static object should have a persistence probability higher than 0.5, we can obtain the static map only containing the objects that exceed this value, as shown in Figure 6.4 (d). The objects that the robot would consider static after including the complete set of mapping sessions are: the 3 sofas, the plant and three chairs, which corresponds to the elements that were not moved during the experiments.

Updating object probabilities during 20 mapping sessions results in a polarization between the objects that have not being detected in most of the sessions (movable objects) and those that remain for almost all the sessions (static objects). As shown in Table 6.2, object movability is scaled in a realistic fashion for an office environment and the robot has effectively learned which objects are more movable (higher values of

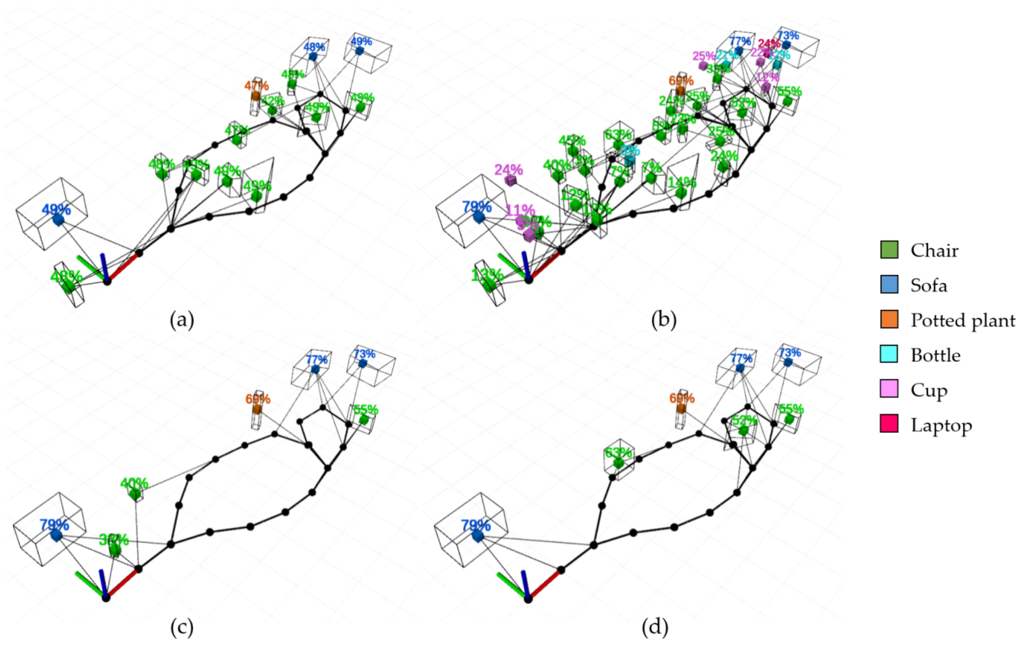


Figure 6.4 Results after 20 mapping sessions and the adaptation in the environment. Figures in (a) and (b) show the resulting map for the first and last mapping sessions respectively, (c) shows the active elements for the last mapping session and (d) shows the objects that are learned as static after the 20 mapping sessions.

movability). The evolution of movability within all the mapping sessions is shown in Figure 6.5.

Table 6.2 Movability according to object class.

Object class	Movability
Bottle	0.8765
Cup	0.8302
Laptop	0.7516
Chair	0.7343
Plant	0.3007
Sofa	0.2319

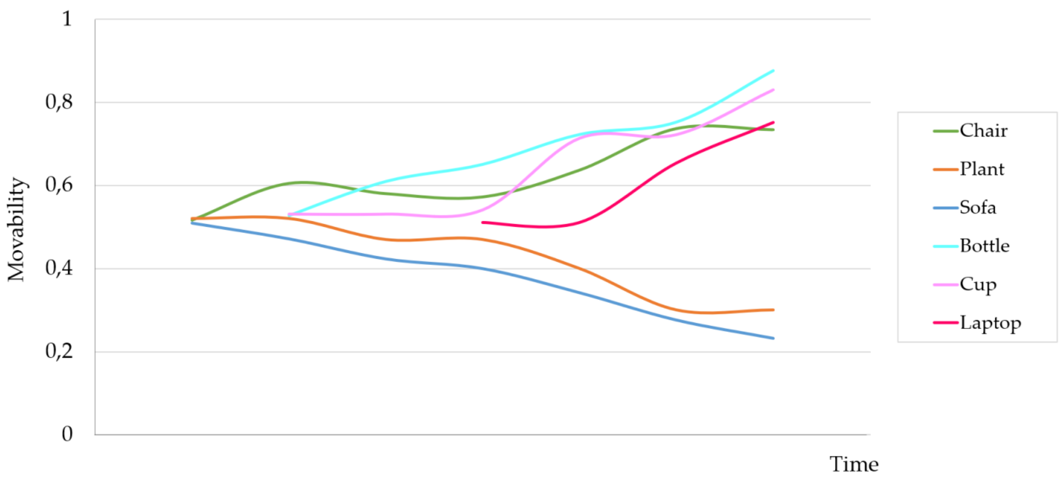


Figure 6.5 Evolution of the movability for each object class during the mapping sessions.

6.3.4 Comparison to Binary Classification Methods

Comparison to binary object classification is included to further evaluate the performance of the method. Binary object classification identifies objects as movable or static. Most of the approaches that use binary classification are meant to map the static objects and discard the movable ones [112, 167, 26]. In order to replicate this behavior, only the objects that have been labeled as active or unknown for all of the mapping sessions are included in the resulting map since they are supposed to be static. Figure 6.6 (a) and (b) show the resulting static map for binary classification and the static map of the proposed method after evaluating the complete set of mapping sessions, respectively.

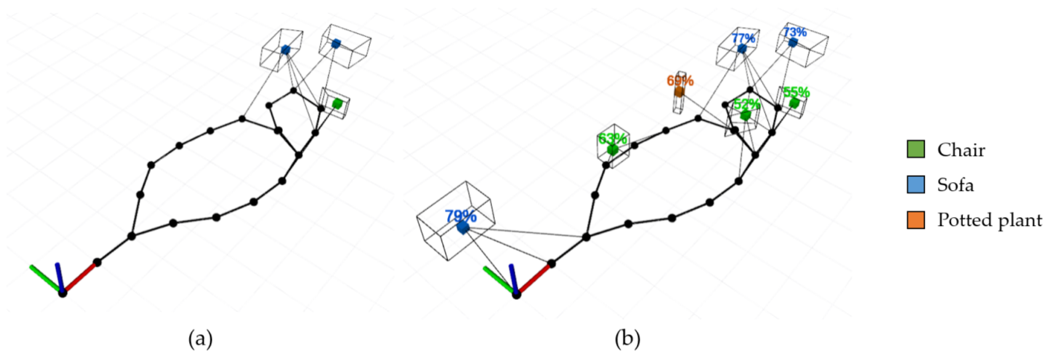


Figure 6.6 Comparison of object classification according to the movability of objects between the binary classification method (a) and using the proposed method (b) after evaluating all the mapping sessions.

As shown in Figure 6.6, the binary classification identifies three objects as static (2 sofas and a chair) and our method identifies 7 objects as static (3 sofas, a plant and 3 chairs). Binary classification overlooks static objects just because they were not detected in one mapping session. An error in the detection is propagated to the classification, like the case of the third sofa and the plant, that were not moved during the mapping sessions. Binary classification obtained a 42.85% of successful static objects mapped in contrast to the 100% of the method proposed. Therefore, robustness is increased in our method thanks to employing the proposed object movability calculation in object classification. In addition, binary classification can just infer about static and movable objects, but it does not give any insight in the degree of movability of the objects.

6.3.5 Discussion

Quantitative results of the proposed method and comparison to binary classification have been presented in this experimental evaluation. Comparison to other methods that define the movability of the environment elements is not appropriate given that those works map the environment using features instead of objects. Although both methods pursue the same objective, as they do not use equivalent information, methods based on feature descriptors cannot be applied to objects. Therefore, here we provide an extensive discussion on how our method presents a contribution in the light of these prior works.

Feature-based approaches [10, 31, 114] gather information from the salient regions of images, scans or point clouds. Features can be merged, removed or assigned a probability that could be registered for dynamic environments, but the meaning of these features is not easy to transfer to the real world. Features represent an abstraction level that allows to know which regions of the environment are prone to changes, but they need a second step to determine which elements are located in that region to gain environment understanding. In contrast, object-based approaches implicitly provide environment understanding, since changes are directly associated to objects. They also share the advantages of feature-based approaches, as they determine the regions of the environment that change more.

Although it was not possible to compare to any other specific method, the movability of objects can be calculated with other well-known methods, such as Bayesian filtering [132]. Here we briefly discuss the comparison of our method with a standard Kalman filter [42]. Kalman filters can estimate the belief of a specific object remaining static or being movable. As proposed for our method, object class movability can be calculated by grouping the objects probabilities (or beliefs) for each object class. Kalman filter and our method can be compared through the results obtained regarding object class movability as shown in Figure 6.7. Our method (orange) and three instances of Kalman filter (blue) are compared for a static object and a highly movable object. The process model for the Kalman filter (defined according to [153] Section 3.2) is initialized with $\mu_0 = 0.5$ and $\Sigma_0 = 0.2$, and it is assumed to be static if measurements are not received ($\mu_t = \mu_{t-1}$ and $\Sigma_t = \Sigma_{t-1}$). The measurement model is defined by each new observation and the measurement noise covariance, R_t. The results show that

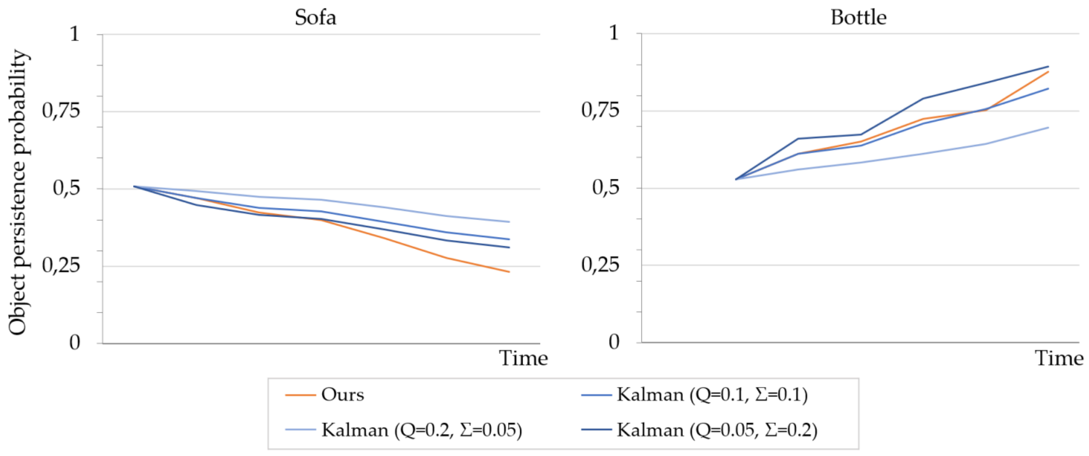


Figure 6.7 Comparison of object movability between the Kalman filter and our method for a sofa and a bottle.

noisier measurements (higher R_t) lead to a slow evolution of object movability, being more difficult to distinguish between static and movable objects. On the contrary, more precise measurements (lower R_t) lead to a more polarized estimation of object movability, especially for movable objects. Our method performs similarly to a Kalman filter of $R_t = 0.2$ for increasing movability (bottle). However, our performance for static objects is increased (sofa), since the system infers faster that the object is static. For these reasons, we conclude that our method performs better than a simple Kalman filter for the task of object movability estimation.

Some advantages can also be found regarding the resulting map. In our method, object probability and active objects are maintained through the different mapping sessions resulting in an improvement compared to other works. Active elements for each mapping session are included, as for [99, 25]. Also the static and dynamic maps of the environment, as for [163, 68]. Comparing the different representations, we can say that the resulting map for the proposed method gives more complete and representative information of the environment than other state-of-the-art methods.

7

Localization and Path Planning in the Hybrid Map

Localization and path planning are two of the key tasks that a map representation enables and they play an important role in robot autonomy. Both tasks have been extensively researched by the robotics community from a topological, metric or combined perspective. In this chapter, we present localization and path planning methods based on the proposed hybrid map.

Localization is the process of knowing and updating continuously a robot position with regard to a map of the environment based on sensor information. Localization is key for autonomous navigation in an environment as otherwise the robot can get lost or move in wrong directions. The localization problem highly depends on the knowledge available initially. Three types of problems are distinguishable according to the difficulty of the problem [153]: position tracking, global localization and the kidnapped robot problem. Position tracking assumes that the initial robot pose is known and consists in robustly tracking that position while the robot moves. In global localization, the initial pose of the robot is unknown and the robot has to identify and track the most promising areas in the environment until a convergence is found. Finally, the kidnapped robot problem is a variation of global localization in which the robot can be moved to other point in the environment and it has to detect and overcome this change.

Once the robot has a good estimate of its pose in the environment, it can perform higher level tasks whose performance depend on that estimation. An example of these tasks is path planning, that consists in calculating the path to reach a goal from the estimated current robot position. Path planning should take into account the global path that drives the robot from the current position to the goal position and local perceptions to avoid colliding with walls and other obstacles. In addition, it could also notice changes in the terrain and other factors that may affect the performance of robot path planning. In order to deal with a global path and local perceptions that may vary the path, path planning has been classically approached as twofold: global path planning using the map of the environment that determines a first path to follow and local path planning that includes variation to the path according to the local perceptions of the robot.

In this chapter, we are presenting a topological localization (position tracking) method that combines information from different components of the hybrid map and a fine-grained metric localization (global localization) that uses the 3D metric maps to estimate robot pose in non-static environments. Regarding path-planning, we are presenting a hybrid global path planning method that combines information from several components of the hybrid map and also different path planning strategies.

7.1 Topological Localization in the Hybrid Map

Topological localization refers to the process of knowing and updating a robot position within a topological map structure. For topological localization, nodes of the topological map correspond to the possible robot poses. Therefore, pose estimation will be as coarse as sparse the topological map is. Denser topological representation will lead to a more geometrically accurate position, however, our aim for topological localization is different from geometric accuracy. For us, it is important to have a topological estimation of the robot location in addition to the metric one (as it is presented in Section 7.2) because topological localization refers to a node that corresponds to a distinguishable area of the environment where the robot will be able to identify different elements and act differently. For example, in the coarsest level, topological localization could serve to distinguish in which room the robot is. If the robot is in a corridor, a living room or a study room, it could act differently independently of its metric pose.

Our topological localization is approached as a position tracking problem in which the initial position (node) of the robot is known prior to localization. Then, the purpose of the algorithm is to robustly track the robot while it moves in the environment. As we will see later, we have also developed a fine-grained metric localization that could provide the initial pose for the topological localization, and the later will be mainly used when the robot is moving through planned paths in the environment.

Our hybrid map contains several topological maps and knowing where the robot is in the different components decreases the localization uncertainty and may help in situations where the robot may get lost. In addition, external semantic information such as scene recognition can be used to enhance the performance of certain components.

In this section, we present an approach to robot topological localization in static environments based on the hybrid map proposed in this thesis. The system exploits topological, semantic and geometric information in order to estimate the most likely nodes in different components of the hybrid map. In addition, the estimations obtained for some components are then used to improve the estimation in other components.

We approach topological localization from a probabilistic perspective based on hidden Markov models (HMMs). A HMM is a probabilistic Markov process in which the states to estimate are unknown or hidden. Its aim is to estimate the distribution of probability of the hidden states based on the available information, which is the previous state of the system and its observable outputs or measurements.

In the following, we are going to briefly summarize the related work regarding topological localization in static environments. Later, we will summarize the components of the hybrid map included in the environment model used for estimation and present in detail the proposed localization method in each component and how components can benefit from each other. We will finish the section about topological localization with an experimental evaluation of the approach.

7.1.1 Related Work

Robot localization, including topological approaches, has been widely researched by the robotics community for the last decades. According to [139], there are three types of approaches to the topological localization problem: structure-based methods that represents the environment as a topological map and predicts robot pose by finding the sequence of observations and transitions that best matches the structure of the map [1, 11, 35, 41, 144, 168, 184]; image-based methods that model the environment as a database of images linked to locations [5, 7, 20, 29, 158, 166]. They use image retrieval techniques to identify the database images most relevant to the query, which are then used to estimate the pose of the robot; and, learning-based methods that represent the scene by a learned model, which either predicts matches for pose estimation or directly regresses the pose [6, 107, 123].

Regarding structure-based approaches that rely on predicting robot pose based on the map of the environment and current transitions and observations, Bayesian filters have been widely used. Cassandra et al. [35] apply a Markov process to represent the belief state of the robot over topological nodes. The observations used in the prediction are walls, doorways, open areas and unknown. HMMs are also used in [144] to infer current robot location by comparing the mapped Generalized Voronoi Diagram (GVD) with the observations. The comparison is performed by a geometric descriptor of the GVD meet points based on emanating edges and their angles and, distance to obstacles. In [184], Zhu et al. exploit the landmark sequences of steady objects using a second order HMM. Common objects are used as landmarks and the occurrence order allow to match the current location of the robot to the topological map. In [1], a simple linear classifier and HMMs are used to determine the floor of a building where the robot is and identify two motion patterns: within-floor motion and motion between floors through WiFi signals. This method defines different transition matrices for the HMM for inter and intra level motion. In [168], pipe network localization includes a measure of the distance traveled in the HMM. The topological map contains a node for each pipe junction and observations are based on distance traveled and the number of exits in the current junction.

Different approaches for image-based methods have been also proposed. Ulrich et al. [158] built a database where images are associated to the topological adjacency map of the rooms of an apartment. Given color images, topological localization consists in identifying the most similar image to the observation. For that purpose, each color band is analyzed separately in oder to determine the best match and a voting scheme is used to select the current location. In [166], a coarse-to-fine strategy for topological localization is presented where global and local features are computed from the input image. Global features are used for coarse localization to select a set of promising locations. Then, local features are used to make the final decision. In [20], sets of color histograms are grouped according to their location. Histogram matching is performed to determine the best match by treating each color band separately.

Some approaches have benefited from the advantages of both, structure-based and image-based methods [12, 50, 102, 121, 172]. In [172], topological localization is first performed through feature matching in order to select promising locations. Then, particle filters are used to precisely determine the robot location within the promising locations through metric information. SeqSLAM++ [121] is based on the well-known method SeqSLAM [115] in which image sequences are compared to a reference image to predict robot location. One of the improvements included in SeqSLAM++ is the combination with a Markov process in order to select a set of promising robot positions before applying image matching. Similarly, in [50] HMM are used to exploit temporal look-ahead by selecting promising candidate images to narrow the best match search.

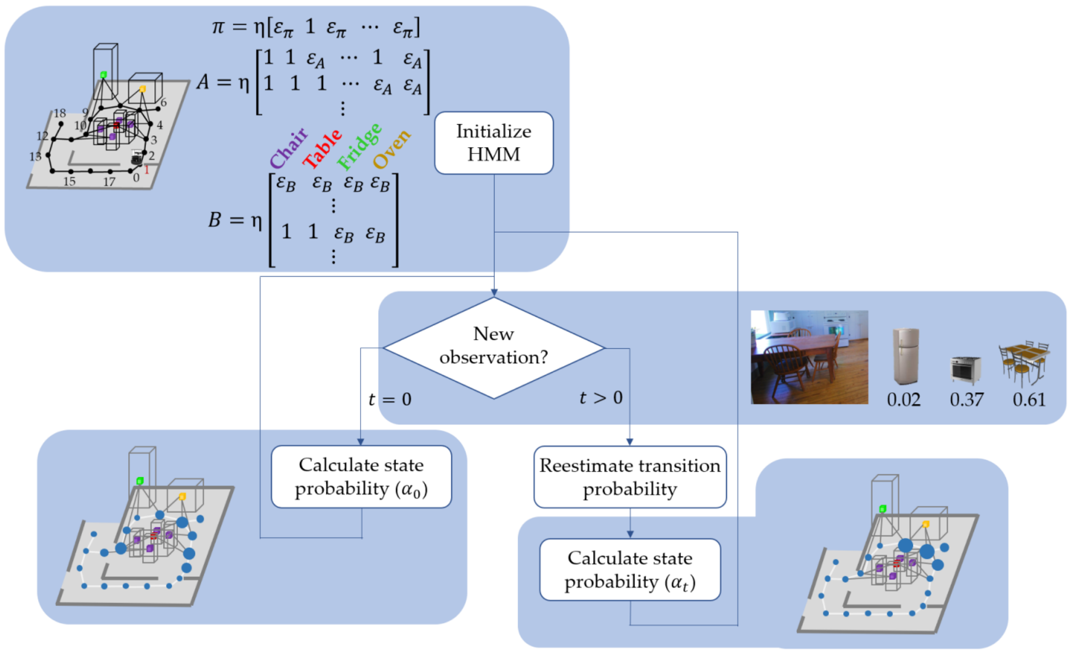
Recently, approaches for topological localization using deep learning have gained importance. In [6], Convolutional Neural Networks (CNN) are used to calculate descriptors for images and perform descriptor matching. Similarly, in [107], CNNs are used for extracting semantic information (objects and rooms) and room inference.

In this section, we present a structure-based approach to identify the most promising location of the robot in two components of the hybrid map through HMMs. Our final result could be considered a coarse-to-fine approach since the estimation of the higher level component (rooms) influences the precise estimation within the rooms. In the same spirit as [168], we use metric information in the prediction process in addition to topological and semantic data.

7.1.2 Localization in the Object-based Pose Graph

Topological localization in the object-based pose graph has been approached using a HMM as state estimation. As explained before, HMM is a probabilistic discrete estimation method that can be considered as the simplest Bayes filter. In this case, we use the observations and transitions of the robot to estimate the most likely node for the robot. Objects are used as observations and an initial observation probability is built from the objects linked to different pose nodes. Transition probability is initially defined according to the connectivity of the pose graph. In addition, we reestimate the transition probabilities after each observation. Figure 7.1 shows the flowchart of the proposed method. First, the HMM is initialized by defining its parameters A, B and π according to the map of the environment (these parameters will be described later in this section). Then, every time that a new observation is received the system calculates the state probability distribution that reflects the most likely pose nodes for the robot location. Observations contain the probability distribution of different objects.

Robot localization requires sensor information to match the current observation of the robot with the nodes of the map. In the same way, the nodes of the topological map have to be linked with the observations registered from them. For this reason we are using the object-based pose graph to estimate the node where the robot is. The object-based pose graph consists of two types of nodes: object nodes and pose nodes.



$\pi = \eta[\varepsilon_\pi \ 1 \ \varepsilon_\pi \ \cdots \ \varepsilon_\pi]$
$A = \eta \begin{bmatrix} 1 & 1 & \varepsilon_A & \cdots & 1 & \varepsilon_A \\ 1 & 1 & 1 & \cdots & \varepsilon_A & \varepsilon_A \\ & & \vdots \end{bmatrix}$
Chair
Table
Fridge
Oven
$B = \eta \begin{bmatrix} \varepsilon_B & \varepsilon_B & \varepsilon_B & \varepsilon_B \\ & \vdots \\ 1 & 1 & \varepsilon_B & \varepsilon_B \\ & \vdots \end{bmatrix}$
Initialize HMM
New observation?
$t = 0$
$t > 0$
0.02
0.37
0.61
Calculate state probability (α_0)
Reestimate transition probability
Calculate state probability (α_t)

Figure 7.1 Flowchart of the topological localization algorithm in the object-based pose graph. First the HMM is initialized according to the information stored in the map (connection between pose nodes and objects detected at each pose node) and the robot known initial pose. Then, every time the robot receives a new observation, where the distribution probability of different objects is given, the system calculates the state probability distribution or forward probability. For the second a subsequent observation a reestimation of the transition probability is performed prior to state estimation. In the state probability estimation step, we have represented the probability of each pose node according to its size. The bigger, the most likely that state is.

Pose nodes refer to different poses of the robot and they are connected between them through feasible paths. In addition, pose nodes are connected to the object nodes that were detected from them. This relation between pose nodes and object nodes allows us to estimate the current pose node given the observations captured by the robot sensors. In this case, the observation captured by the robot consists in RGB-D images from which object entities have been extracted through the object detector ResNet-101 [75].

A localization algorithm based on hidden Markov models is applied on top of the object-based pose graph to determine the probability of being at each node of the map. HMMs are used as they offer a robust and efficient solution for the probabilistic definition of discrete representations. HMM is a probabilistic model used to characterize systems where it is assumed that future states depend only on the current state, not on what occurred before. In a HMM, the system is assumed to have unobserved (hidden) states. Although HMMs have been widely studied, it is one of the most robust algorithms to manage discrete data probabilities, as the case of topological maps.

HMMs are commonly represented using the tuple $\lambda = \{Q, V, A, B, \pi\}$. Q refers to the set of possible states, which we will call individually as x_i. V refers to the set of all possible observed values in each state. A, B and π will be described in detail and refer respectively to the state transition probability distribution, the observation probability distribution, and the initial state probability distribution.

The state transition probability distribution, A or $p(x_t \mid x_{t-1}, u_t, m)$, determines the probability of transiting from a state x_{t-1} to a state x_t given the motion model of the robot u_t and the map of the environment m. For us, states are the pose nodes of the object-based pose graph. The transition probability A is a square matrix in which the number of columns and rows corresponds to the number of pose nodes, N, and it is estimated according to eq. (7.1) based on the existence or non-existence of a connection between state x_i and x_j in the object-based pose graph. If the connection exists $x_j \in \{priors_i\}$, $a_{i,j}$ is assigned to 1 and otherwise it is assigned to a very small value, ε_A. The individual $a_{i,j}$ values of the state transition probability distribution are then normalized so they all add up to 1 through the normalization factor η.

$$A = \begin{bmatrix} a_{0,0} & \cdots & a_{0,n} \\ \vdots & \ddots & \vdots \\ a_{n,0} & \cdots & a_{n,n} \end{bmatrix}; \quad a_{i,j} = \begin{cases} \eta & \text{if } x_j \in \{priors_i\} \\ \eta\, \varepsilon_A & \text{otherwise} \end{cases} . \tag{7.1}$$

The formulation of the state transition probability depends only on the connectivity of the underlying pose graph. We have also proposed a variation of the state transition probability that also depends on a qualitative geometric coefficient based on the expected and real orientations to see the objects from the pose nodes. The difference between the expected orientation and real orientation was used to penalize transitions according to plane quadrants. Small penalization was applied if expected and real orientations belong to adjacent quadrants, large penalization was applied for opposite quadrants and none for same quadrants. Further details of this variation can be found in [69]. This heading penalization, w_h, was then included in the state transition probability:

$$a_{i,j} = \begin{cases} \eta\, w_h & \text{if } x_j \in \{priors_i\} \\ \eta\, \varepsilon_A & \text{otherwise} \end{cases} . \tag{7.2}$$

The observation probability distribution, B or $p(z_t|x_t)$, refers to the probability of observing a certain observation z_t given a state x_t. In this work, observations refer to detected objects. B is a matrix in which the number of rows are the number of object types, M, and the number of columns corresponds to the number of pose nodes, N. The observation probability is estimated according to eq. (7.3) taking into account the objects associated to each pose node of the object-based pose graph. If the object z_k was observed from pose node x_i, $z_k \in \{o_i\}$ where $\{o_i\}$ refers to the set of objects associated to pose node x_i, $b_{z_k,i}$ is assigned to 1. Otherwise, $b_{z_k,i}$ will be assigned to a very small value ε_B. The individual $b_{z_k,i}$ values of the observation probability distribution are then normalized so they all add up to 1 through the normalization factor η.

$$B = \begin{bmatrix} b_{0,0} & \cdots & b_{0,n} \\ \vdots & \ddots & \vdots \\ b_{m,0} & \cdots & b_{m,n} \end{bmatrix} ; \quad b_{z_k,i} = \begin{cases} \eta & \text{if } z_k \in \{o_i\} \\ \eta\,\varepsilon_B & \text{otherwise} \end{cases} . \tag{7.3}$$

The initial state probability distribution, π, determines the probability of being at each state x_i at the beginning of the localization process. As the initial position of the robot is known, 1 will be assigned to the pose node of the initial estimation, $\hat{x_0}$ (current node and first pose node of the path to perform), and a residual value, ε_π, to all other pose nodes:

$$\pi = \begin{bmatrix} \pi_0 & \dots & \pi_n \end{bmatrix} ; \quad \pi_i = \begin{cases} \eta & \text{if } x_i = \hat{x_0} \\ \eta\,\varepsilon_\pi & \text{otherwise} \end{cases} , \tag{7.4}$$

where η represents the normalization factor. If we were dealing with a global localization problem, a uniform distribution would be used for π.

HMMs are the simplest Bayesian filters and can be applied through different implementations. In this work, a simple forward algorithm is used in which the forward probabilities are calculated as the posterior probabilities. Forward probabilities α are calculated as the probability of being at state x_i given the observation z_t, $p(x_i|z_t)$. Forward procedure is iterative and new probabilities are calculated at each time t. Different expressions are used to calculate the forward probability distribution for $t = 0$ and the subsequent estimations since the initial probability distribution π is only used in the first case:

$$\alpha_0(i) = p(x_i|z_0) = \eta\,\pi\,b_{z_0,i}\,p(z_0) \ , \tag{7.5}$$

where η is the normalization factor, $p(z_0)$ the probability of the first received object and $b_{z_0,i}$ the probability of having perceived that object class given each pose node x_i. The probability of the perceived object, $p(z_0)$, is related to the certainty of the detection provided by the object detection method. The probability distribution $\alpha_0(i)$ provides the first estimation for all the pose nodes of the object-based pose graph and the most likely pose node, X_t will correspond to the maximum value obtained in $\alpha_0(i)$ using the the maximum a posteriori (MAP) estimate:

$$X_t = \underset{i}{\operatorname{argmax}}(\alpha_0(i)) \ . \tag{7.6}$$

Forward probabilities for subsequent pose estimations, $t > 0$, depend on the transition probability between two nodes and the prior probability of each node:

$$\alpha_t(i) = p(x_i|z_t) = \eta\,(\sum_{j=0}^{N-1} \alpha_{t-1}(j)\,a_{j,i})\,b_{z_t,i}\,p(z_t) \ , \tag{7.7}$$

where η is the normalization factor, $\alpha_{t-1}(j)$ the forward probability distribution obtained for the previous time step, $a_{j,i}$ the transition probability from node x_j to node x_i, $p(z_t)$ the probability of the received object and $b_{z_t,i}$ the probability of having received that object class given each pose node x_i. As calculated for α_0, the estimated location of the robot will correspond to the one with the largest probability, eq. 7.6.

A reestimation procedure is also implemented in order to improve the transition probability distributions after each observation. The new value for each state transition $a_{i,j}$ is calculated from every previous forward probability $\alpha_{t-1}(j)$, its prior state transition probability $a_{i,j}$ and the probability of receiving the observation that occurred in that state, $b_{z_t,j}$:

$$New\ a_{i,j} = \eta \sum_{j=0}^{N-1} \alpha_{t-1}(j)\,a_{i,j}\,b_{z_t,j} \ , \tag{7.8}$$

where η refers to the normalization factor of the probability distribution.

Reestimation of the state transition probability adjusts the method to the current state of the robot given the last observation and the probability of having that observation at each state. As we will see later, the reestimation of the transition matrix has a positive impact in the final state estimation.

The explained method allows us to estimate the robot location regarding the pose nodes of the object-based pose graph. In the following section, we will see how this method can help in the estimation of the robot location in other components of the hybrid map. Results will be then shown, both, for this method on its own and the joint method on the hybrid map.

7.1.3 Localization in the Topological Map

Localization in the topological map consists in determining the most likely room in the environment given robot observations. As a reminder of the topological map, it consists of supernodes (rooms) that are connected through transitions (doors). Each supernode is associated with a class or scene type and all the elements of the other components that belong to that supernodes: objects, poses, etc. In this sense, we approach localization in the topological map as a combination of two of its main attributes: scenes and poses. Figure 7.2 shows the flowchart for topological localization in the topological map. Supernode probability is computed through the estimation of the scene type and the most likely pose nodes given an observation. The observed scene type is compared to the scene type stored for each supernode and the probability of the pose nodes is grouped according to the supernode they belong to.

Through a scene recognition algorithm [77] we can estimate the probability for each scene type given the current observations of the robot. Given that each supernode has been previously associated with a scene type, comparing the sensed scene with the stored ones for each supernode gives a first estimate of the most likely supernodes for the robot location.

In addition, as we have seen in the previous section, we already have some information of the probability of finding the robot in each pose node of the object-based pose graph. As we know which pose nodes belong to each supernode, we can accumulate the probability of each single pose node according to supernodes. This will lead to a new probability distribution for the supernode, $p(s_i \mid x)$, that is built according to:

$$p(s_i \mid x) = \sum_{j=0}^{N-1} p(\dot{x}_i \mid z_t) \quad , \tag{7.9}$$

where $\dot{x}_i$ refers to the pose nodes that belong to supernode s_i.

We can merge scene probability, $p(s_i \mid \xi)$ where ξ represents the probability distribution obtained for the sensed scene, and pose nodes probability, $p(s_i \mid x)$, for each supernode in order to obtain the probability distribution of the supernodes:

$$p(s_i) = \eta \, p(s_i \mid x) \, p(s_i \mid \xi) \quad , \tag{7.10}$$

where η is the normalization factor between all the supernodes probabilities.

The new enhanced probability distribution for supernodes can be propagated back to the object-based pose graph level:

$$p(x_i \mid z_t, s_i) = \frac{p(x_i \mid z_t)\, p(s_i)}{p(s_i \mid x)} \quad , \tag{7.11}$$

where $p(x_i \mid z_t, s_i)$ refers to the pose node probability once the supernode information is added; $p(x_i \mid z_t)$ refers to the original pose node probability distribution also referred as $\alpha(i)$; $p(s_i)$ refers to the final supernode probability distribution and $p(s_i \mid x)$ to the supernode probability distribution only taking pose node information (not scene information) into account.

This operation leads to an improvement also in the estimation of the pose node where the robot is. Propagation maintains the probability distribution between nodes of the same supernode but strengthens the nodes that belong to the most probable supernode.

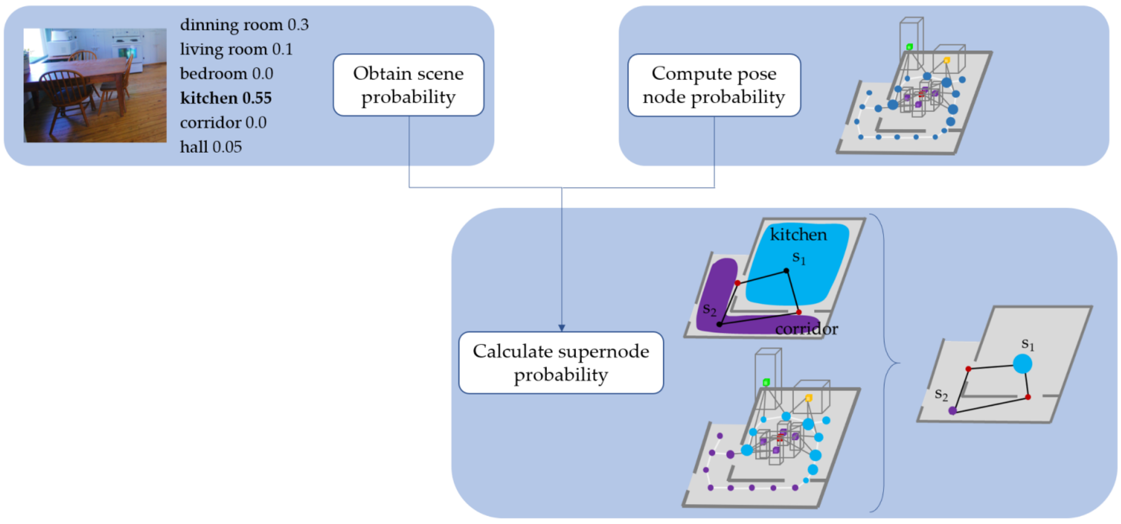


Figure 7.2 Flowchart of the topological localization algorithm in the topological map. Scene probability distribution from a scene recognition method and pose node probability distribution is combined to estimate supernode probability. Given the observed scene and the scene stored in each supernode from one side and the probability of each individual pose node and the pose nodes that belong to each supernode, we can estimate the most likely supernode for the robot location. Blue is used for the supernode and pose nodes that belong to the kitchen and purple to the supernode and pose nodes that belong to the corridor. The size of supernode probabilities represent how likely they are (the bigger, the most likely).

In this section, we have first seen the mathematical development to compute supernode probability from pose node and scene probabilities. Afterwards, we have shown how the computed supernode probability can be used to update the original pose node probability. In the following, we see the experimental evaluation of the proposed method.

7.1.4 Experimental Evaluation

We present experiments to evaluate both topological localization methods. First, the topological localization algorithm for estimating the robot pose in the object-based pose graph is presented. Since this method includes different steps and evolved during the time of this thesis, the experiments show the original method and posterior modifications. Then, the combined method for topological localization in the topological map and the object-based pose graph will be evaluated. We will show the improvements both in the topological map and the object-based pose graph due to the proposed method.

Experimental Setup

The experiments in the object-based pose graph have been performed in a quasi-realistic manner, as the robot was teleoperated through the environment while it was estimating its pose. It is quasi-realistic because the observations of the robot were provided by the user. On the contrary, experiments in the hybrid map were performed in a real-world environment where the robot was teleoperated while the localization and perception modules were autonomously running. Both experiments were performed using Mob-E (a differential drive robot built at our lab), Figure 7.3 (left). For the experiments in the hybrid map, Mob-E was equipped with an Asus Xtion Pro camera for perception.

Topological localization experiments were performed and published in a previous version of the object-based pose graph that was called observation adjacency graph. For this reason, the map presented in the following results differs slightly from the object-based pose graph presented in Chapter 3. As a reminder, the object-based pose graph consisted of pose nodes and object nodes. Pose nodes defined the trajectory traveled by the robot and object nodes were connected to the pose nodes from where the object was observed. On the other hand, the observation adjacency graph consisted of object nodes that were connected to each other based on their adjacency. We decided to modify the observation adjacency graph and convert it into a pose graph due to the complexity in autonomously building the observation adjacency graph and the

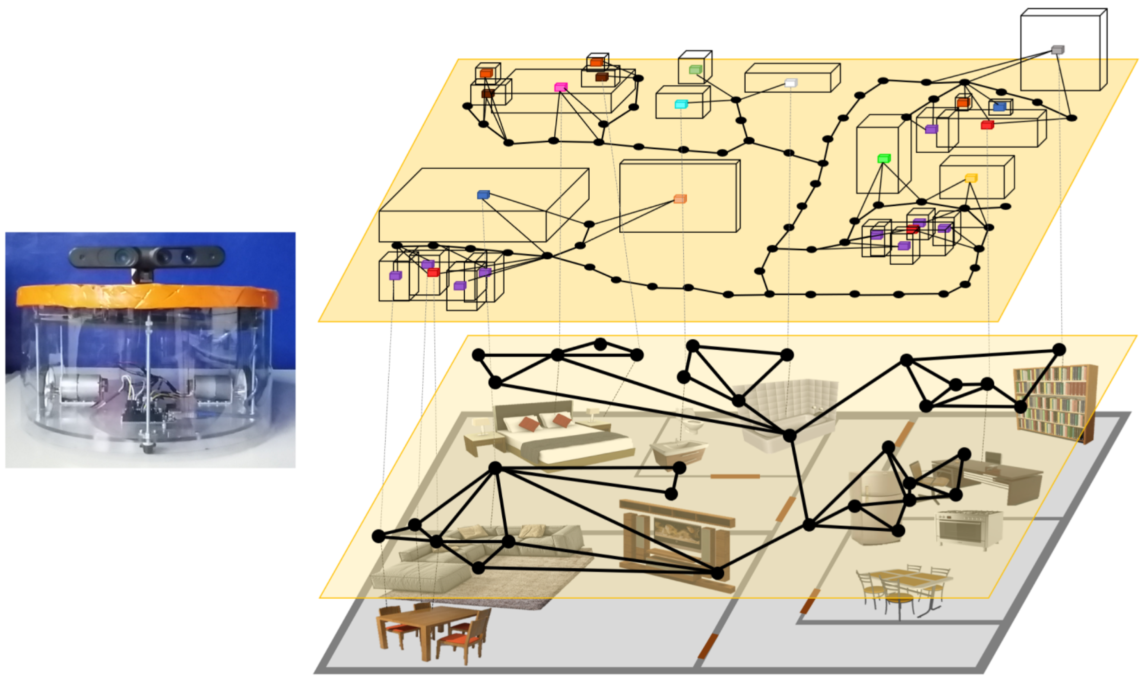


Figure 7.3 Mob-E mobile robot is shown in the left, as it is the robot with which we performed the experiments. In the right, the correspondence between the object-based pose graph and the observation adjacency graph is shown. In addition, the environment and the observation adjacency graph shown are the ones used in the first topological localization evaluation.

arbitrariness of the connections between nodes. Both graphs and the environment that they represent are shown in Figure 7.3 (right). The object-based pose graph is shown on top of the observation adjacency graph.

Evaluation of Localization in the Object-based Pose Graph

Experiments consisted in moving the robot through the environment and capture the localization results according to the received observations. The impact of the reestimation of the transition matrix and the inclusion of the qualitative geometric coefficient, w_h, is studied in two paths in the environment. The sequence of observations for these paths are shown in Figure 7.4, one traversing the living room and the other one moving from the bedroom to the bathroom.

To test the failures of the localization system and its recovery capacity, several errors were intentionally introduced within the observations. The evaluated localization cases are:

- correct observations according to the path.
- missing one observation within the path.
- repeating one observation within the path.
- including one extra observation.

In addition, observations are provided as a probability distribution between the different possible objects. None observation used in these experiments corresponds 100% to one observation type as we try to simulate the uncertainty found in real perception systems.

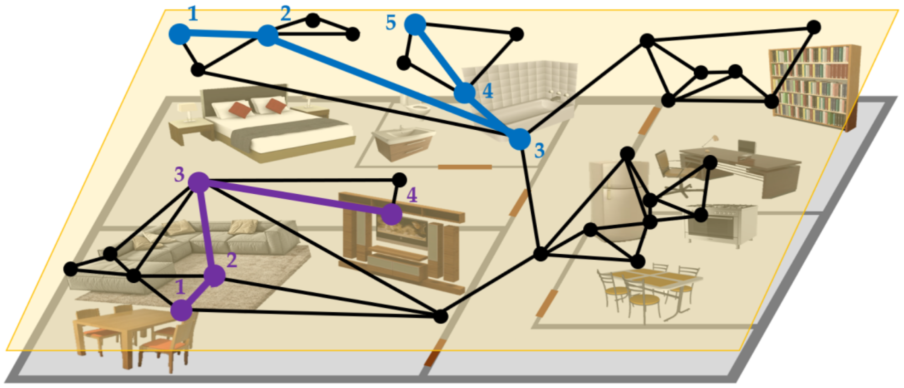


Figure 7.4 Sequence of observations for the two evaluated paths. Blue for the bedroom path and purple for the living room path.

First, we want to study the impact of the reestimation of the transition matrix and the inclusion of the geometric coefficient in the calculation of the probability distribution along a path. For this, we compare the probability of the ground-truth node in the three situations: original, with reestimation and with reestimation and geometric information. Figure 7.5 shows the probability distribution along the living-room path. Each group of three columns refers to one estimation and each of the columns refers to the original estimation, including reestimation of the transition matrix and adding geometric information, respectively. In addition, in Table 7.1, the quantitative results for the two paths and the overall improvement compared to the original method, and the method with reestimation is presented.

As observed in the results, the probability for the ground-truth node always increases when using geometric information and reestimation of the transition matrix, meaning that the system is more certain that the robot is located in the right position. It is

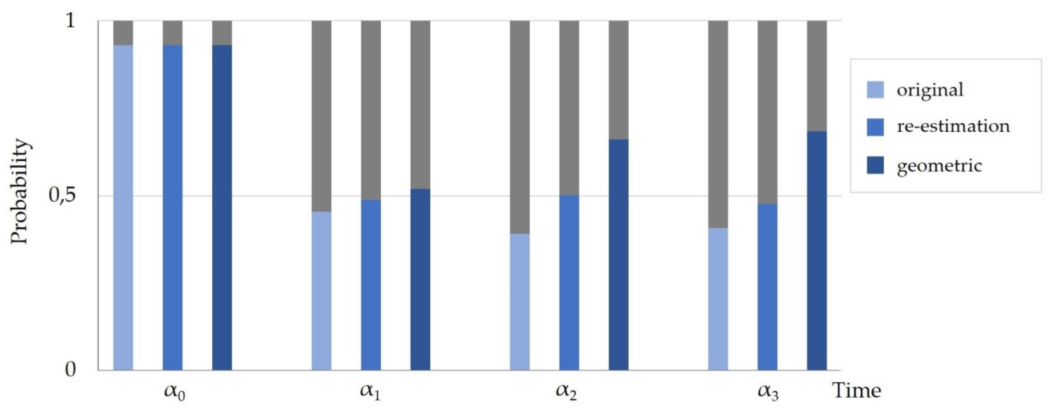


Figure 7.5 Probability distribution for four observations along the living room path. Light blue represents the original probability of the ground-truth node, medium blue the probability including reestimation of the transition matrix and dark blue the probability including the reestimation and geometric information. Gray is used to represent the probability of all the other nodes.

Table 7.1 Probability results for the correct nodes of living room and bedroom paths when correctly executed.

Test		α_0	α_1	α_2	α_3	α_4
Living room	original	0.9301	0.4536	0.3911	0.4071	
	reestimation	0.9301	0.4874	0.5004	0.4765	
	geometric	0.9301	0.5192	0.6613	0.6845	
improvement		-	14.46%	69.08%	68.14%	
Bedroom	original	0.9854	0.9592	0.8874	0.5765	0.7783
	reestimation	0.9854	0.9689	0.9032	0.6016	0.8720
	geometric	0.9854	0.9693	0.9083	0.6389	0.9213
improvement		-	1.05%	2.35%	10.76%	18.37%

worth mentioning the transitions between α_0 and α_2 for the living room path and the transition between α_2 and α_3 for the bedroom path. In the first case, the two first observations received by the robot are chairs and the initial position of the robot is surrounded by four chairs which will produce four different probability modes. As two of the chairs are further, they will receive a lower probability and for this reason the initial probability is not 0.25, as could be expected, but 0.45. In addition, since the heading of the robot coincides with the required heading for the ground-truth nodes, the estimation using geometric information highly improves the prediction. In the second case, the observation received by the robot is a door and the robot is surrounded by two doors, the door to the corridor and the door to the bathroom. This leads to a reduction of the confidence in the ground-truth node, however with the next observation (a WC) the uncertainty is highly reduced again.

Up to now, the results presented concerned only the correct execution of the path and sequence of observations. In the following, in order to test the robustness of the system against detection errors, we are showing the results for the four mentioned cases (correct execution, missing observation, repeating observation and including observation). The results for performing these tests for the three evaluated methods, original (or.), with reestimation (re.) and with reestimation and geometric information (ge.), are included in Table 7.2. The results refer to the % of right goal probability, this means that the last goal of the path is correctly estimated although the robot might have gotten lost in between, notice that the higher the percentage the more robust the system is; the % of middle error probability, this refers to bad probability estimations in nodes that are not the goal of the path, notice that the lower the percentage the more robust the system is; and the % of recovery, which refers to amount the middle nodes error that the system has been able to overcome, notice that the higher the value the more robust against failures the system is. If the localization system has not

Table 7.2 Overall probability results for the different types of tests for living room and bedroom paths.

Test		% right goal probability			% middle error probability			% recovery		
		or.	re.	ge.	or.	re.	ge.	or.	re.	ge.
Living room	correct	100	100	100	0	0	0	-	-	-
	miss	61.5	66.67	66.67	47.35	44.44	22.22	67.5	75	50
	repetition	95.67	100	100	25.4	20	6.67	97.3	100	100
	inclusion	59.31	66.67	100	27.9	20	20	61.33	66.67	100
	global	**79.12**	**83.33**	**91.66**	**25.16**	**21**	**12.16**	**75.37**	**80.55**	**83.33**
improvement			15.85			51.67			10.56	
Bedroom	correct	100	100	100	0	0	0	-	-	-
	miss	96.67	100	100	28.22	25	18.75	93.25	100	100
	repetition	87.33	100	100	26.67	12.5	0	85.67	100	-
	inclusion	66.67	100	100	20	16.67	12.5	70.51	100	100
	global	**87.66**	**100**	**100**	**18.72**	**13.54**	**7.81**	**83.14**	**100**	**100**
improvement			14.07		58.28			20.28		

miscalculated the probability for any middle node, the % of recovery is shown as "-" because there was no need of recovery.

The most robust results were obtained for the localization method including geometric information and reestimation of the transition matrix as they obtained the highest percentage of right goal estimations in both paths, the lowest percentage of middle error estimations and the highest percentage of recovery after error.

With this experiment, we validate the positive effect that the reestimation of the transition matrix and the inclusion of geometric heading information has in the estimation of the location of a mobile robot in a topological map based on objects. In the following experiment, we combine these results with localization in a higher level map in order to observe the effect that the estimation of each level has in the other and vice versa.

Evaluation of Localization in the Hybrid Map: Object-based Pose Graph and Topological Map

Real-world experiments consisted in moving the robot through the environment while it captures the objects observed and predicts its current location based on those observations and the movement of the robot. In this evaluation, we want to show the localization results both in the preliminary version of the object-based pose graph (the observation adjacency graph) and the topological map. We will focus our evaluation in comparison of supernode and node estimation if the information of the other component is taken into account or not.

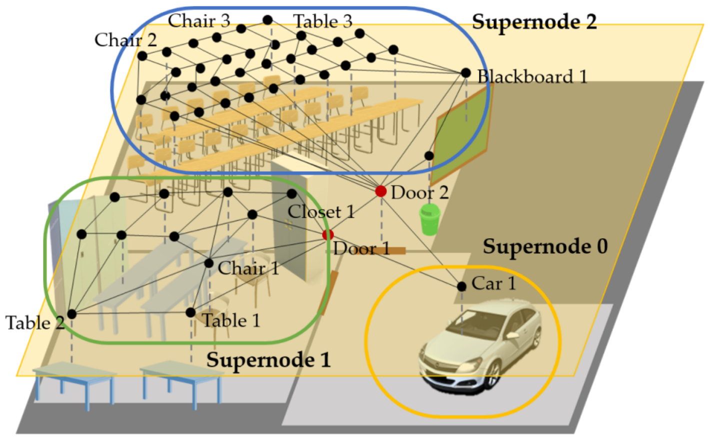


Figure 7.6 Environment for topological localization evaluation. The environment consists of three supernodes and multiple nodes that belong to those supernodes.

This experiment was performed at the University Carlos III of Madrid and consisted of three rooms as shown in Figure 7.6: a classroom, a laboratory and a garage. In Figure 7.6, it is shown the topological map of the environment and the observation adjacency graph. Regarding the topological map, three different rooms are shown (each identified with a color, blue for classroom, green for laboratory and yellow for garage) connected through the transition nodes or doors marked in red. The observation adjacency graph is also shown for the environment and we can see the nodes that belong to each supernode.

First, we evaluate the influence of the inclusion of node information in the supernode probability. As stated before, we are calculating supernode probability in a twofold: using semantic scene information and combining the original node prediction of the nodes that belong to each supernode. Figure 7.7 shows the probability distribution for the supernodes while the robot moves in the environment. Each color is used for a supernode type and the ground-truth supernode is included along the prediction. Two columns are included for each prediction where the first column refers to the prediction just using semantic scene information and the second one for the prediction including also node estimation. Since the probability of the ground-truth supernode increases at every iteration, the inclusion of node estimation leads to an improvement in the certainty of the supernode prediction. It is remarkable that scene classification errors are overcome, as the laboratory which at some instants was misclassified as a classroom.

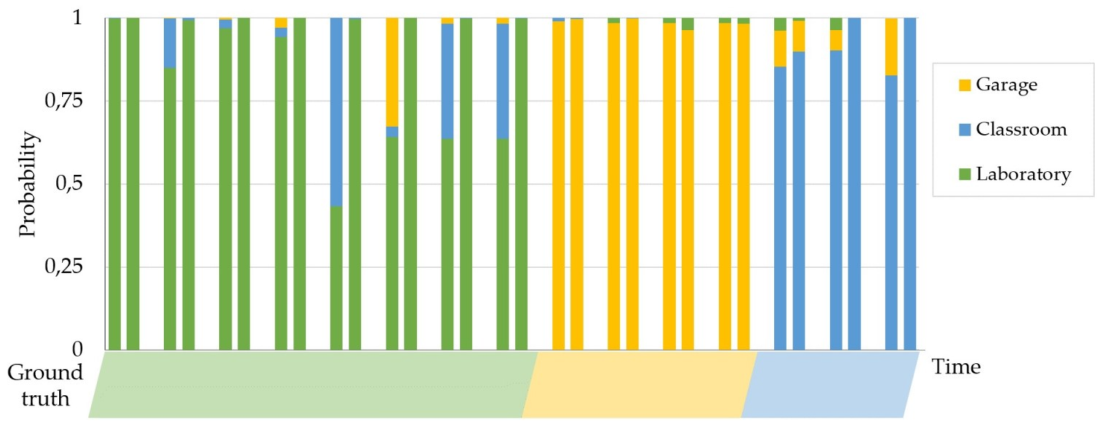


Figure 7.7 Probability distribution for supernode estimation using only scene information and including also node estimation in the observation adjacency graph. Yellow is used to represent garage probability, blue for classroom probability and green for laboratory probability. The ground-truth supernode over time is also included in the graph.

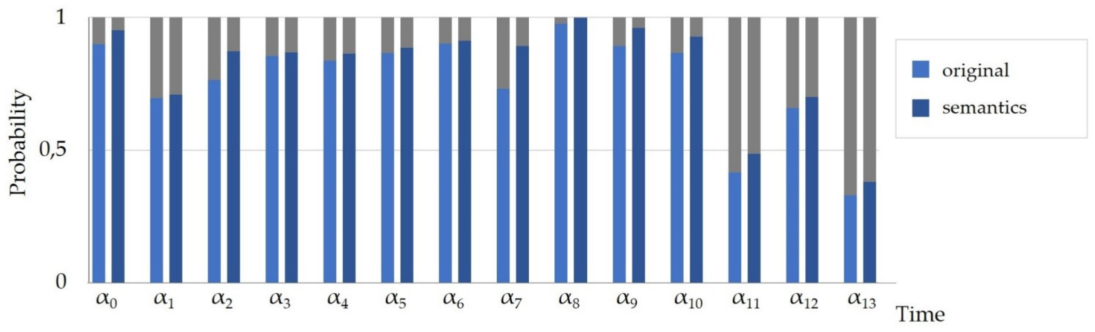


Figure 7.8 Probability distribution for nodes of the observation adjacency graph. The original probability (light blue) is compared to the probability after backpropagation of supernode information (dark blue).

In addition, we want to evaluate the effect that the improved supernode estimation has in single node prediction. As described before, we propagate the improvement of the supernode probability to the nodes that belong to that supernode. Figure 7.8 shows the probability distribution of node estimation overtime. The estimation for the ground truth node is represented in blue and gray corresponds to all the other nodes. In the first column, in light blue, the original node probability is included and, in the second column, in dark blue, the probability estimation after backpropagating the supernode estimation is shown. The average improvement in node estimation including backpropagation of supernode estimation for the ground-truth node probability is 7.72 %.

With this experiment we validate that topological localization in a hybrid map can be improved thanks to the combination of the estimation in different components of the map. Especially, we show how the estimation of supernodes (rooms) can benefit from the estimation of individual nodes and individual nodes can benefit from the estimation of the supernode.

7.2 Metric Localization in Non-static Environments

Fine-grained and precise localization is an essential capability for a mobile robot operating in real-world environments since it represents a key factor for the autonomy of the robot. Despite being researched for several decades, robot localization in the real world still has many challenges, especially if the world is not static. Dynamic environments are affected by moving and movable objects, things that can change the appearance, as well as the change caused by external factors such as lighting or seasonal changes. All these conditions increase the difficulty in localization.

Among the changes that affect dynamic environments, we focus in this thesis on objects that can be moved or may change their appearance. We will refer to these environments as non-static. Changes in non-static environments can be challenging for robot localization as the internal representation of the environment that the robot has built no longer matches the current state of the world. This problem can cause the robot to make wrong pose estimations or even get lost.

In this section, we present an approach to global robot localization in non-static indoor environments but also static environments can benefit from our proposed method. The system exploits metric and semantic information for pose estimation in environments in which objects are movable or can change in their geometric appearance. We use an environment representation based on a truncated signed distance field augmented with semantic information estimated by the mobile robot. For localization in the environment, we use a particle filter to estimate the pose of the robot.

In the following we are going to briefly summarize the related work regarding metric localization exploiting semantic information in non-static environments. Later, we will summarize the components of the hybrid map included in the environment model used for localization and present in detail the proposed pose estimation and the experimental evaluation of the approach.

7.2.1 Related Work

Several works have previously used semantics for metric visual localization in both, static and dynamic environments. The inclusion of semantic information is useful in static environments as it helps in pose disambiguation [181, 15]. In the work by Bavle et al. [15], object semantics are included in probabilistic localization by comparing the average semantic distance of a given object in the image with the mapped elements.

In the context of localization in dynamic and non-static environments, approaches that use semantics in indoor environments have mainly focused on dealing with high dynamics such as people moving in the environment [170, 73, 124, 103, 171, 175]. These approaches work in small environments where the robot or camera is always facing to the same scenario and only small camera movements occur. Most approaches eliminate dynamic elements and rely only on the static parts for pose estimation. In the works by Xiao et al. [170] and Xu et al. [171], in addition to the localization rejecting dynamic elements, a semantic map of the static elements is built. On the contrary, approaches such as CubeSLAM [175] take into account dynamic elements to improve pose estimation. CubeSLAM generates cuboids around objects and uses bundle adjustment to optimize camera and objects poses. Cuboid dimensions and semantic types are used to compute measurement errors and improve camera pose estimation. In addition, the semantics of static objects are also used in the estimation process.

Works dealing with low dynamics, such as objects that are moved or change in their appearance, have been mainly approached from a feature-based perspective [126, 45, 49]. In the work by Patel et al. [126], semantically enhanced features are used for pose estimation. Images are firstly semantically segmented in an object level and then, features are extracted for each detected object. Features can only be matched if they belong to the same semantic type. Other works such as the work by Dayoub et al. [45] and Derner et al. [49] assign weights to each feature according to their stability in the environment. In the work presented by Stachniss and Burgard [146], laser information is used to estimate robot pose in a non-static environment through a Rao-Blackwellized particle filter. In that work, the robot collects non-static information of the environment and pose estimation takes into account several configurations of the non-static parts of the environment computed through clustering.

Localization in outdoor dynamic environments have been approached from multiple perspectives: appearance-based methods that exploit image sequences [161, 162] or navigation sequences [115] and methods that exploit semantics [147, 39]. Our work is more similar to the later, especially localization systems where robot observations are semantically labeled and both, static and movable elements are used for localization [147, 155, 131, 37, 39]. In the work by Toft et al. [155], localization combines feature matching and semantic information. Similarly to Patel et al. [126], images are semantically labeled and each feature is assigned a semantic type. They generate camera pose hypotheses and assign a score to them according to the number of semantically-labeled matches. Stenborg et al. [147] proposes a particle filter based on semantic localization in which each pixel in the image and 3D point in the map are compared according to their semantic type. In addition of using semantic information to filter dynamic elements, Chen et al. [39] integrate the semantic information into a surfel-based map in order to improve ICP matching in the context of SLAM.

In our work, we present a probabilistic localization algorithm for indoor environments that includes semantic information in the estimation process, similarly to [15, 147]. Unlike most approaches for non-static indoor environments [45, 49, 126], we exploit semantic segmentation for every pixel and 3D point in the environment. However, we share with Dayoub et al. [45] and Derner et al. [49] a weighting method in the estimation process. In the case of the previous works, each feature is weighted according to its stability. In our case, we implement a weighting method between metric and semantic information, which can be associated with the persistence of objects.

7.2.2 Pose Estimation in Non-static Environments

Given an initial environment model, our approach estimates the robot pose from depth and semantic information obtained from an RGB-D sensor. To this end, we use a representation based on our hybrid map in which a whole truncated signed distance field (TSDF) augmented with semantic information is implemented. Pose estimation exploits directly the TSDF and its semantics. In this work semantics have been manually added to the TSDF, however, given today's semantic segmentation systems, we could directly turn an RGB or RGB-D image into a semantically segmented image in real time, for example, using Bonnetal [116].

The environment representation for this work is a global TSDF of the environment augmented with semantic information. In order to build the global TSDF, we are using the work of Zeng et al. [178]. As an extension to their work, we modify the approach through the inclusion of a semantic value for each voxel. The semantic value indicates the class of object that the voxel belongs to and it can only be assigned for surface voxels.

The global TSDF will correspond to the merge of all the 3D submaps of our hybrid map. In addition, the efficiency of the method would be highly increased if we considered the structure of the hybrid map in the pose estimation (as the metric search would be bounded to the room size instead of searching in the whole environment). For this work, we decided to keep pose estimation in a global level for simplicity in the algorithm. However, in a future line, we could implement a method to estimate pose probability in several submaps and, after convergence, stick with the submap of the highest probability.

Regarding localization, we use a particle filter [48] to estimate the pose of a ground robot as they naturally deal with multiple hypotheses. Particle filters calculate a belief over the robot pose using a set of weighted particles and each particle represents a candidate robot pose. First, the particle filter predicts the pose of the robot based on the odometry obtained from its wheel encoders. Then, a weight is assigned to each particle that represents how well the surroundings of the particle match with the robot observation. Finally, a new set of particles is created from the resampling of the old ones, where the chances of survival of a particle is proportional to its weight.

We use a standard odometry motion model to predict the new pose of the particles. The odometry motion model uses odometry measurements obtained by the wheel encoders of the robot to calculate the pose increment between x_{t-1} and x_t. Given that increment, we calculate the new distribution of the particles, $p(x_t^{[k]} \mid u_t, x_{t-1}^{[k]})$ according to the model by Thrun et al. [153] where u_t refers to the motion information.

We propose an observation model that takes into account depth and semantic information of the environment to compute the particles weights. Each observation obtained from the RGB-D sensor consists of a depth image and a color image. We are transforming color images into semantic images by manually labeling each observation. Through the observation model we can correct the weight of each particle so it refers to how accurately that particle matches to the observation. In Figure 7.9, the global flowchart of the correction step is included. Our correction step evaluates every pixel of the observation for every particle. Firstly, the 3D pose x and corresponding voxel of the pixel is calculated by backprojecting the pixel $q = \begin{bmatrix} a & b \end{bmatrix}^\top$ that has a depth $D(q)$:

$$\mathrm{x} = \begin{bmatrix} \frac{a-c_x}{f_x} D(q) \\ \frac{b-c_y}{f_y} D(q) \\ D(q) \end{bmatrix} , \tag{7.12}$$

where a and b are the pixel 2D coordinates in the image and c_x, c_y, f_x and f_y are the intrinsic parameters of the camera, assuming a pinhole camera model.

Then, the corresponding voxel and the pixel itself are compared in order to calculate the pixel probability. Pixel probability depends on semantic and metric differences as will be explained later. When all the pixels have been evaluated, the particle weight is computed. This process iterates for all the particles in the particle filter so we can then resample the particles and identify the most promising areas for the pose estimation.

From a more formal perspective, through the observation model, we update the particle weight w each time a new observation z_t is received taking into account the pose of each particle $x_t^{[k]}$ and the map of the environment m. We calculate the weight of the particle as the product of the probability obtained for each individual pixel i of the observation z_t, where N represents the total number of pixels in the observation:

$$w^{[k]} = \eta \, p(z_t \mid x_t^{[k]}, m) = \eta \prod_{i=0}^{N-1} p(z_t^i \mid x_t^{[k]}, m) , \tag{7.13}$$

the weights of the particles are then normalized so they all add up to 1. The constant η represents the normalization factor.

In a similar spirit as beam models for range finders [65] where the observation model for a single beam is a mixture of four densities (see [153], page 157, eq. (6.12)), we calculate the probability for every pixel as the mixture of two probabilities: $p_{sdf}(z_t^i \mid x_t^{[k]}, m)$

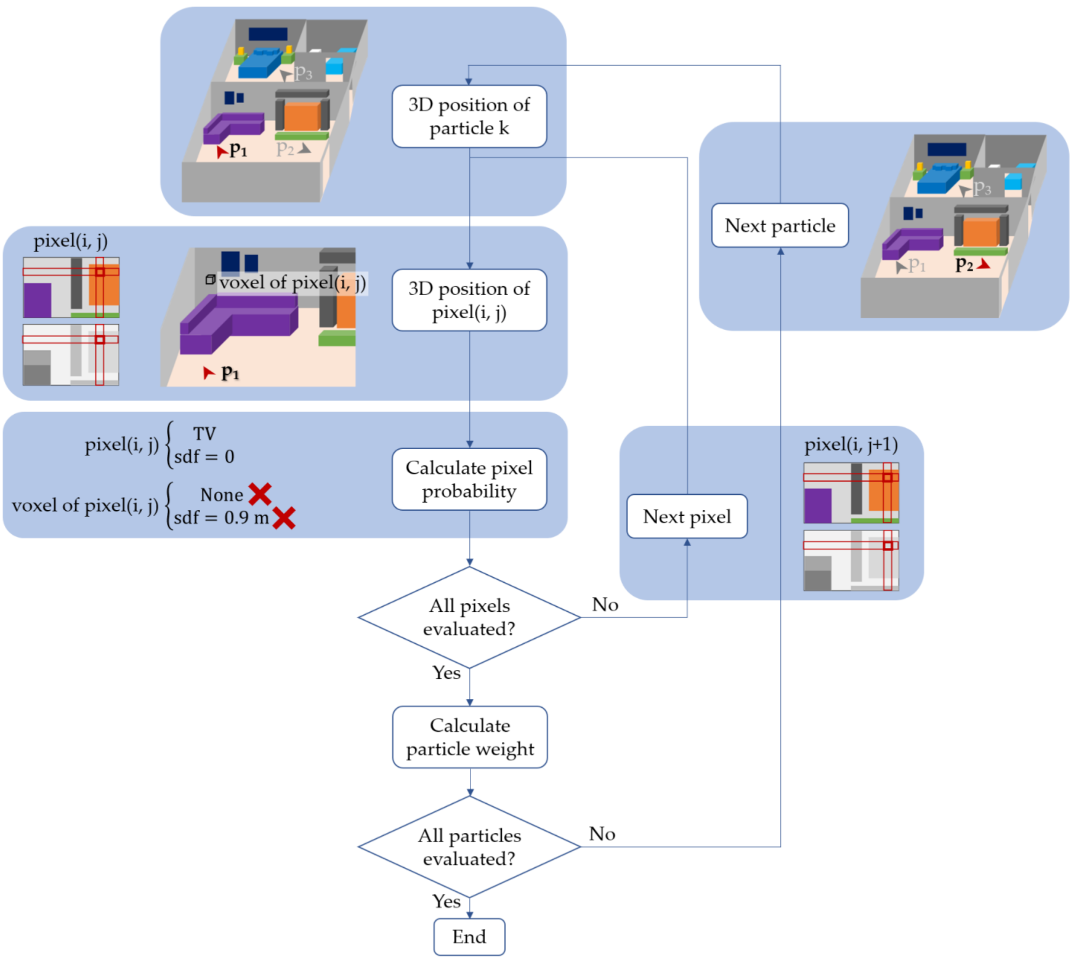


Figure 7.9 Flowchart of the proposed metric localization method. The system evaluates each pixel of the observation for every generated particle. Given the particle pose, the 3D position of the pixel is computed and the resulting corresponding voxel is compared to the pixel itself. That pixel probability is calculated based on the semantic and metric differences between the corresponding voxel and the pixel. This process iterates and calculates the particle weight for every pixel and every particle.

accounting for metric information and $p_{sem}(z_t^i \mid x_t^{[k]}, m)$ for semantic information. The two different probabilities are mixed by a weighted average defined by the weighting factors z_{sdf} and z_{sem} with $z_{sdf} + z_{sem} = 1$. We calculate the observation model at a pixel level as:

$$p(z_t^i \mid x_t^{[k]}, m) = \begin{pmatrix} z_{sdf} \\ z_{sem} \end{pmatrix}^\top \begin{pmatrix} p_{sdf}(z_t^i \mid x_t^{[k]}, m) \\ p_{sem}(z_t^i \mid x_t^{[k]}, m) \end{pmatrix}, \tag{7.14}$$

with i referring to each pixel of the sensor data.

As mentioned before, we directly exploit the TSDF to calculate the metric probability $p_{sdf}(z_t^i \mid x_t^{[k]}, m)$, since it directly represents the distance of the point to the nearest surface. Given the 3D point obtained for each pixel i, we can calculate its corresponding voxel in the model representation. The sdf value of the corresponding voxel is retrieved, denoted as sdf_i. The distribution $p_{sdf}(z_t^i \mid x_t^{[k]}, m)$ has a maximum at $sdf_i = 0$, what means that the point is already a surface. The variable σ_{sdf} represents the standard deviation for metric probability. We compute the likelihood for the metric information as:

$$p_{sdf}(z_t^i \mid x_t^{[k]}, m) = \exp\left(-\frac{sdf_i^2}{2\sigma_{sdf}^2}\right). \tag{7.15}$$

For the likelihood given the semantic information, we introduce the term semantic distance S_i. Given a pixel with a certain semantic class, the semantic distance refers to the Euclidean distance between the position of the 3D projection of that pixel and the position of the closest voxel in the model representation that belongs to the same semantic class. Similarly as before, $p_{sem}(z_t^i \mid x_t^{[k]}, m)$ takes the maximum value for $S_i = 0$, which means that the corresponding voxel for that point already belongs to the desired semantic class. The standard deviation for semantic probability is denoted by σ_{sem}. We calculate the likelihood of the semantic cue as:

$$p_{sem}(z_t^i | x_t^{[k]}, m) = \exp\left(-\frac{S_i^2}{2\sigma_{sem}^2}\right). \tag{7.16}$$

After correcting particle weights, we resample the particles and otherwise run a standard Monte-Carlo localization approach [48].

7.2.3 Experimental Evaluation

Experimental Setup

We have performed the experiments in a real world environment using a Turtlebot 2 robot and an Asus Xtion Pro RGB-D sensor. Figure 7.10 shows the resulting TSDF augmented with semantic information that will be used as model representation for the experiments.

Regarding the weights of the observation model, we give equal importance to metric and semantic information, $z_{sdf} = z_{sem} = 0.5$. Another promising approach could be varying the weight according to the change in geometric appearance for different types

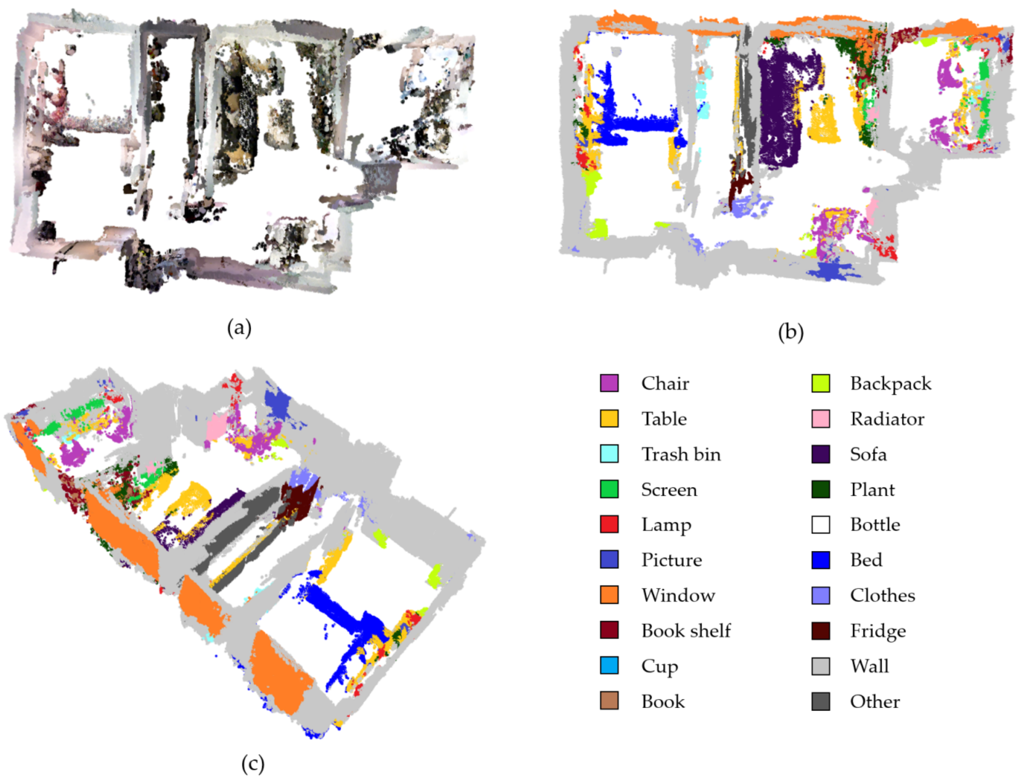


Figure 7.10 TSDF with semantic information of the real world used for the experiments: (a) shows an illustrative view of the TSDF with RGB information, (b) and (c) two views of the TSDF with semantic information. Color-coded semantics corresponds to the included table.

of objects or based on the semantic class of the object. In that case, weights could be calculated as $z_{sdf} = 1 - \mu$ and $z_{sem} = \mu$ where μ represents the change in geometric appearance of the object class, as calculated in Chapter 6.

Our evaluation includes comparisons to MCL taking only metric and only semantic information into account. For metric-only estimation, we assign $z_{sdf} = 1$ and $z_{sem} = 0$. On the contrary, for semantic-only estimation, we use $z_{sdf} = 0$ and $z_{sem} = 1$.

Performance in Static Environments

The first experiment evaluates the performance of our approach in static environments to support that the combination of metric and semantic information improves robot localization even in static environments. For this experiment, we run the localization algorithm for different paths without including any change in the environment compared to the initial TSDF.

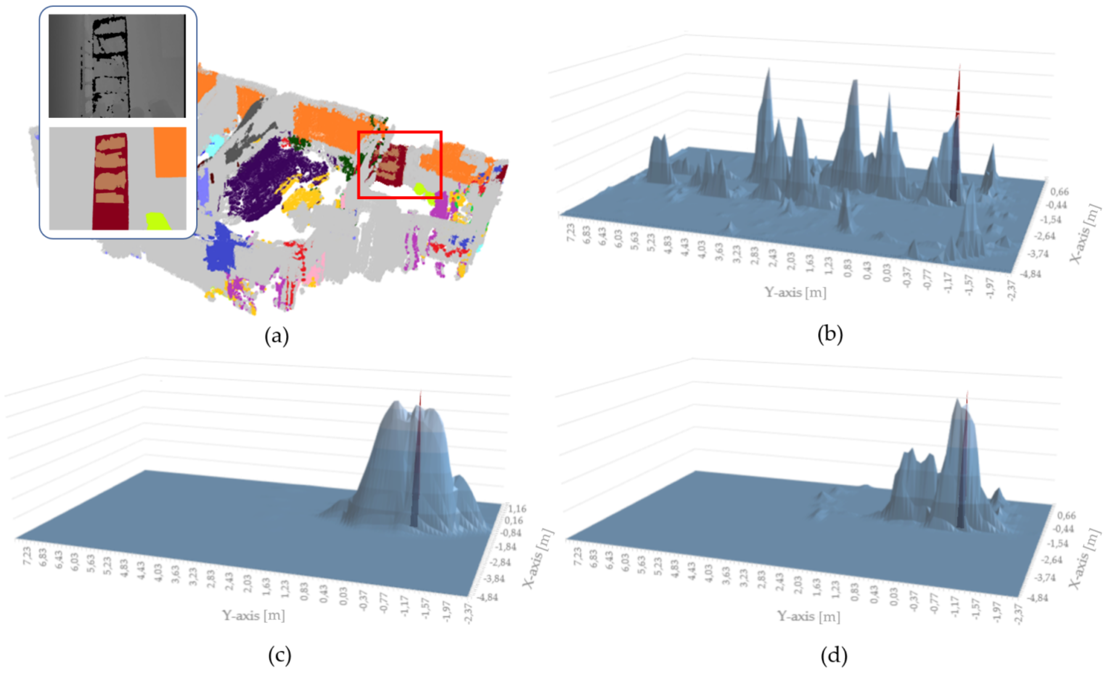


Figure 7.11 Result for the estimation given a single observation in the static environment: (a) shows the depth and semantic images for the observation and the TSDF of the environment with a red square highlighting the observed area; (b), (c) and (d) show the estimation for metric information, semantic information and the combination of both, respectively. The ground truth is represented with a red column.

First of all, we show how the estimation of the pose for a single observation varies when considering only metric information, only semantic information, and the combination of both. Given the observation shown in Figure 7.11 (a), the metric-based estimation is shown in Figure 7.11 (b), the semantic-based estimation in Figure 7.11 (c) and the estimation combining both sources of information in Figure 7.11 (d). For this experiment, we evaluated the observation model on a dense 3D grid at every 10 cm and $\pi/8$ rad. For each of the grid positions, we are representing in Figure 7.11 the maximum weight of the orientations evaluated. The ground truth position is shown in red. The metric-based model is multimodal as many places in the environment could match the proposed observation. The semantic-based model identifies two modes that correspond to the two book shelves with books that are in the environment. Finally, the combined estimation benefits from both models and the belief is peaked around the ground truth for the observation.

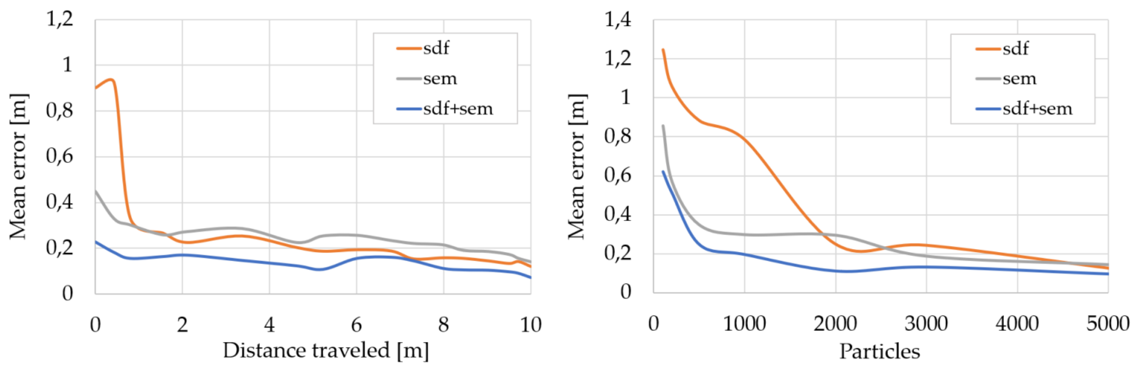


Figure 7.12 Quantitative evaluation of localization in static environment. In the left, position error (m) w.r.t. distance traveled in the static environment for the three studied cases: metric (sdf), semantic (sem) and combined (sdf+sem). In the right, position error (m) w.r.t. number of particles in the particle filter for the same three studied cases.

Results for a path in the initial environment show quantitatively how the position error differs between the three cases: metric (sdf), semantic (sem) and combined (sdf+sem). We have initialized the particle filter with 3000 particles and run the experiment 5 times in order to obtain average results. Figure 7.12 (left) shows how the mean error evolves while the robot executes the path. Once the filter has converged, our approach (sdf+sem) obtains an average error of 13.1 cm with a variance of 1.2 cm. The average results after convergence are summarized in Tab. 7.3. The estimation using only metric information fails to estimate the pose in one execution and thus it obtained high error values. If the erroneous execution is overlooked the mean error would be 0.24 m, still being twice as large as for our approach. Figure 7.12 (right) shows the evolution of position error as we vary the number of particles involved in the estimation. Our approach requires less than 1000 particles to obtain an average result of 0.2 m, whereas metric-based and semantic-based estimation require approximately 4000 and 3000 particles, respectively.

Table 7.3 Mean error and variance in static environment.

Method	Mean error [m]	Variance [cm]
sdf	0.4371	59.31
sem	0.1894	2.50
sdf+sem	0.1309	1.23

Performance in Non-static Environments

The next set of experiments has been conducted to support that our approach successfully deals with changes in the environment thanks to the combination of metric and semantic information. For this experiment, we run the localization algorithm for different paths of approximately 10 m of length in different configurations of the environment. Different configurations involve objects that changed in their appearance or objects that were moved (removing mapped objects and adding new objects). Some examples of these changes are shown in Figure 7.13.

First of all, we want to evaluate the convergence capability with movable objects of each of the evaluated cases: metric (sdf), semantic (sem) and our approach (sdf+sem). For this experiment, we have initialized the particle filter with 3000 particles for the three cases and we have executed 3 times each of the 4 different paths that are involved in this experiment. Figure 7.14 (left) illustrates how the position error evolves while the robot moves. We can distinguish an initialization phase until the distance traveled

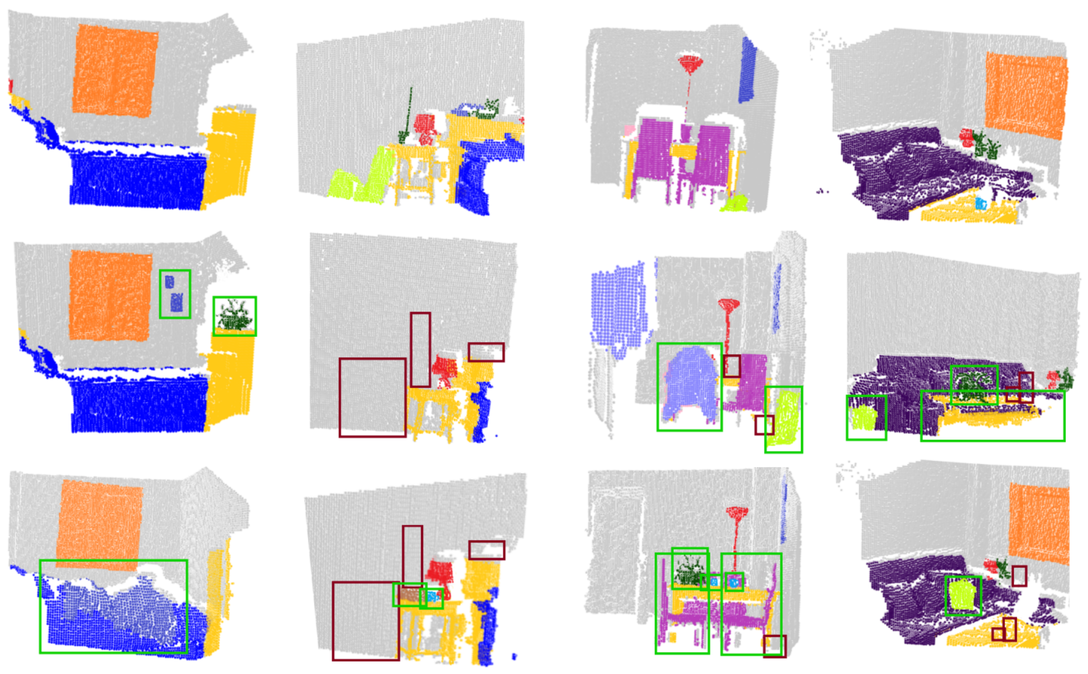


Figure 7.13 Examples of changes in the environment. The first row corresponds to scenes in the static environment. Second and third row corresponds to different configurations of the environment including changes. Objects that are no longer in the scene are marked in red and objects that are new or have changed in their appearance are marked in green.

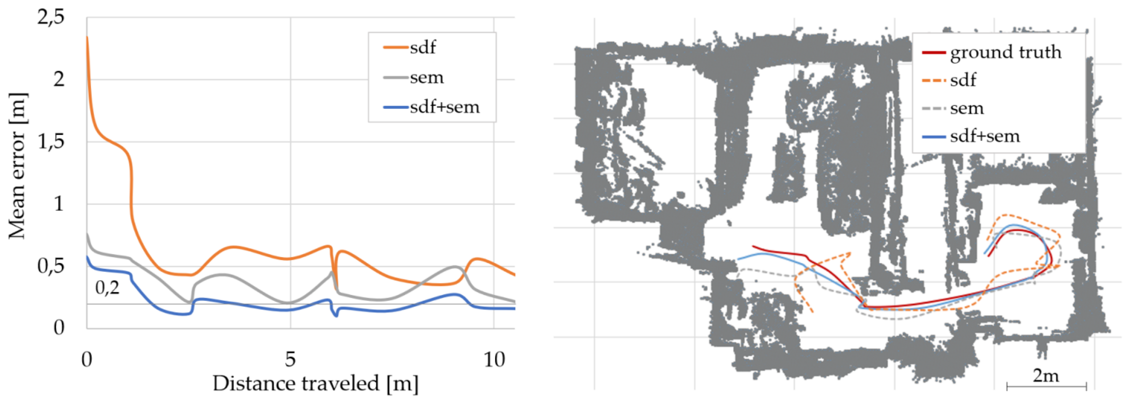


Figure 7.14 Quantitative evaluation of localization in non-static environment. In the left, position error (m) w.r.t. distance traveled in non-static environment for the three studied cases: metric (sdf), semantic (sem) and combined (sdf+sem). In the right, the pose estimation for a path in the environment with changes is shown. Ground truth and the estimation for the three studied cases is included.

is approximately 1.75 m. During pose tracking, we can observe in the three methods some increases in mean position error due to the non-static objects and changes. Our approach (shown in blue) obtains a lower mean error during the execution of the paths and it is more robust to changes as it always keeps a mean error close to 0.2 m. It is worth mentioning the sharp change in error that takes place at a distance traveled of approximately 6 m. At that point, the robot was about to enter in the last room. First, it was observing an uncertain scene due to some changes, then entering the room the robot faced the door (which offered geometrically consistent information) and the uncertainty reduced. Once inside the last room, the new changes that the robot was perceiving increase the uncertainty again. Figure 7.14 (right) shows the estimation for one of the recorded paths. Our approach (blue) is the first to converge to the ground truth and it successfully tracks robot pose along the path. Table 7.4 shows average position errors for the three cases after convergence. If we compare these results with the ones obtained in the static environment, the mean position error for our approach increases 1.46 times, for semantic-only it increases 1.92 times and for metric-only it increases 2.16 times. In addition, mean position error for our approach in the non-static environment is lower than metric-only error in the static environment. This suggests that appearance changes in the environment have a low impact on our method exploiting geometric and semantic information jointly.

Table 7.4 Mean error and variance in environment with changes.

Method	Mean error [m]	Variance [cm]
sdf	0.5269	16.93
sem	0.365	7.31
sdf+sem	0.1912	2.47

In summary, our evaluation suggests that our method provides competitive localization with increased accuracy in non-static indoor environments. At the same time, our method upholds the statement that the combination of metric and semantic information is a key for real-world operation in static and non-static environments. As mentioned at the beginning of this section, the efficiency of the proposed localization method could be improved thanks to the hybrid map and it is something that we will approach in the future.

7.3 Path Planning in the Hybrid Map

Hybrid mapping enables more efficient and versatile path planning strategies as the information of different components of the hybrid map can be merged in the planning stage or we can select the most suitable information for different planning strategies. As an example, path planning in a hybrid map could serve to plan a path for an object search task where the robot is told to move to the door of a certain room in order to look for an specific object. In an under-review work, we propose a solution to this problem based on the structure of the hybrid map proposed in this thesis. As the reader might remember, our hybrid map relates places with supernodes in the topological map and the edges of the topological map refer to the transitions between places or supernodes. Since we are referring to an indoor environment, places simplify to rooms and transitions between places to doors. In addition, the traversability graph also includes the transition nodes. So, given the room to reach we can plan a path using the traversability graph to obtain the optimal path to the door of that room. In this simple path planning example, we use the traversability graph to perform path planning but we require the topological map to define the precise goal to reach the entrance of the goal room.

In the following, we are going to briefly summarize the related work regarding path planning in hybrid maps and hybrid path planning. Later, we will see a more complex approach for hybrid path planning in the hybrid map and the experimental evaluation of the proposed approach.

7.3.1 Related Work

Traditionally, hybrid path planning refers to approaches that tackle both global and local path planning simultaneously. As we mentioned previously, global path planning consists of finding the path that connects the starting position and the goal position using an environment representation or map. In addition, local path planning deals with perceiving the surroundings of the robot to make decisions of where to move next. Considering only a global approach, path planning will not be robust against changes in the environment [51, 182]. On the contrary, using only a local approach, path planning is prone to lead the robot to local minima and not reaching the goal [51, 182]. Therefore, the combination of both approaches has been a traditional solution for planning trajectories in environments with changes and neglecting the effect of local minima. This traditional approach for hybrid path planning is mainly based on computing a global path that the robot will follow while the local planner can vary the path to increase the safety of the robot.

Some authors [51, 164, 177] benefit of global and local path planners by combining A* method and potential fields method. A* algorithm generates the set of subgoals to reach the goal and potential fields follows the pre-planned path smoothing the trajectory and avoiding unknown obstacles. Similarly, in [40, 182], authors use A* algorithm for global path planning and dynamic window approach for local path planning in order to track the global path and avoid dynamic obstacles. In [47], path planning with limited sensors capabilities is proposed. When the robot has sufficient information to plan a global path, A* method computes the path and tangential escape algorithm is used as local planner to overcome unknown obstacles. However, if there is a lack of information, tangential escape algorithm will be run on its own avoiding obstacles until there is enough information to plan a global path.

Thrun [151] proposed a hybrid metric-topological map in which path planning was performed on the metric level or the topological level independently. For topological path planning, Thrun proposed a graph search algorithm called value iteration. The path was later optimized using triplet planning in order to generate feasible paths. In this case, triplet planning is not a local planner since it does not use online information of the environment but it has the same aim as it of a local planner.

Other authors have referred to hybrid path planning as the combination of different path planning strategies based on the information of the environment or the task to perform. Differently from the previously mentioned approaches, in this case, the combination does not consist of a global approach and a local one. Two global or two local approaches can be combined to improve the performance of the robot. Foskey et al. [64] propose a method based on generalized Voronoi algorithm in which a path is search in the graph. Afterwards, every path segment is checked and in case of being invalid (contains obstacles or gets too close to them) that segment is replaced with a randomized planning towards the beginning of the next segment. In SmartPATH [36], ant colony optimization and genetic algorithms are combined for solving the global path planning problem. Firstly, optimal paths are calculated using ant colony optimization. These paths are used as initial population for the genetic algorithm obtaining an improvement in search space and time. Mac et al. [110] also propose an optimized global path planning. In their case, Dijkstra algorithm is used to find a first collision-free path and particle swarm optimization is applied to the pre-planned path in order to generate a global optimal path with the focus on minimizing path length and maximizing path smoothness. In [104], multi-resolution lattices are generated to model the environment. High-resolution lattices are used in the vicinity of the robot and the goal, whereas low-resolution lattices are used elsewhere. A modified version of A* algorithm is used to plan paths in both lattice resolutions.

Finally, other works have also taken advantage of hybrid maps to perform more efficient path planning. These approaches use different representations of the hybrid map to contribute to the path planning strategy. Konolige et al. [90] presented an approach to plan trajectories in a hybrid metric-topological map. The map consists of a global topological map and local occupancy grids. In their work, the overall plan is formed on the topological graph but the robot is commanded metrically within the local grid. Dijkstra algorithm is used to compute the global path using the topological graph and ROS navigation stack using A* algorithm computes the local path within a given local grid. In [165], Wang et al. propose a three-level map containing a scene level, a feature level and a metric level. A coarse-to-fine strategy is applied to compute the path. A first global path is computed using Dijkstra algorithm in the scene level. The features covered by the path can guide the movement of the robot using the feature level. As several features are observable from different positions, this path is optimized using again Dijkstra algorithm in the feature level. Finally, the metric path is computed following the pre-planned path and optimizing the turning points for the robot.

The approach that we are proposing in this chapter could be classified in the last group, where hybrid maps are exploited to improve path planning performance.

Differently from the methods mentioned above, our approach alternates between two well-known path planning methods augmented with a local path planning algorithm. Previous works, [90] [165], used the different components of the hybrid map to define a coarse-to-fine path planning strategy. In our case, we take advantage of the components of the hybrid map to define different path planning strategies. In the same spirit as [104], we sought for a coarser path planning strategy when the robot is far from the goal and finer when it is approaching the goal.

7.3.2 Hybrid Path Planning

Hybrid maps and hybrid path planning strategies allow to optimize robot trajectories in terms of their performance and computation. We propose a hybrid path planning method that uses low-resolution topological information for traveling long distances and high-resolution metric information for precisely approaching the goal. As low-resolution topological information we are using the traversability graph and as high-resolution metric information the 3D submaps. In addition, we are using the information of doors contained in the topological map to know when to change from one component to the other. The traversability graph is used until the door of the room where the goal is and the 3D submap of the room is used from the door of the room to the goal.

We refer to this method as hybrid path planning because we also combine path planning strategies. A search-based strategy is used with the traversability graph and a sample-based strategy with the 3D submap. In the first stage, when the robot has to travel a long distance to reach the door of the goal room, we prioritize the efficiency and speed of a search-based method against the precision of the sample-based method. On the contrary, to approach the goal we prioritize the precision of a sample-based method. Dijkstra search algorithm [142] is used on the traversability graph as search-based algorithm. Within the goal room, rapidly-exploring random tree (RRT) algorithm [98] is used on the 3D submap as sample-based algorithm. Figure 7.15 shows a graphical description of the hybrid path planning method. the aforementioned behaviour is represented in the graph where Dijkstra search algorithm computes the path from the starting position (blue circle) to the door of the goal room (blue cross) and RRT is used to reach the goal (red cross). In addition, Figure 7.15 includes a final module for the local planer that we have used to smooth robots trajectory.

Given the starting node and the node that corresponds to the door of the goal room, Dijkstra search algorithm obtains the path that connects both nodes with the lowest cost. Dijkstra search algorithm is an uninformed and complete path planning algorithm that will always find a solution to the search problem if there is any and

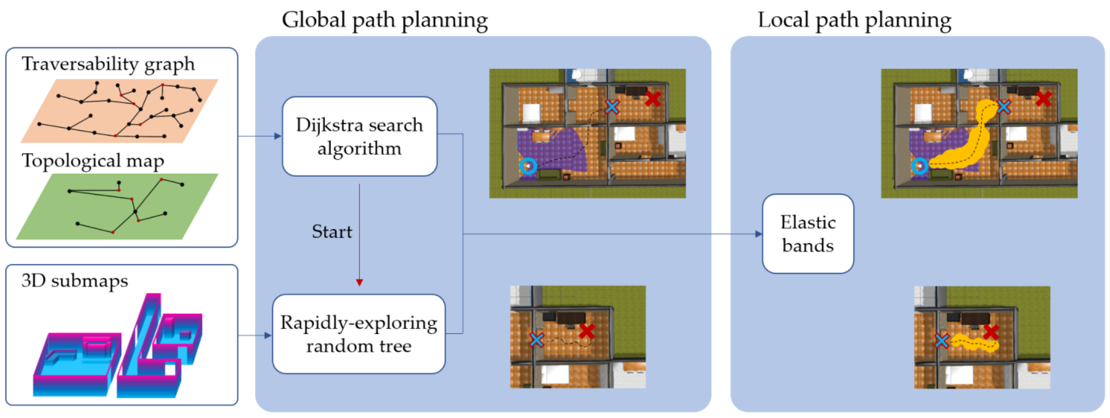


Figure 7.15 Graphical description of the hybrid path planning method. Global path planning consists of a Dijkstra search algorithm using the traversability graph an a Rapidly-exploring random tree (RRT) using the 3D submap. Dijkstra algorithm computes the path from the stating position (blue circle) to the entrance of the goal room (blue cross). From there, RRT calculates the path to reach the goal (red cross). Local path planning consists of Elastic bands algorithm for both of the computed paths.

from the multiple solutions will select the one with the lowest cost. The cost of the path can be associated only with the distance (in that case it will select the shortest path), however it allows to associate the cost to other factors. In our case, we use this cost to avoid paths that have become non traversable.

Dijkstra algorithm modifies the breadth-first strategy [183] by always expanding the lowest-cost node first rather than the lowest-depth node and will explore all nodes until the optimal solution is found. Dijkstra algorithm is an easy to implement search-based method that is efficient for small or medium search spaces. Other more optimal search-based algorithms could be implemented for large or highly-connected environments such as A* [74] or D* [148].

Once the robot is at the entrance of the goal room, the 2D occupancy grid obtained from the 3D submap of the room is used to compute the remaining path to reach the goal through RRT. RRT incrementally constructs a tree from the initial position by connecting random samples spread in the search space. Each sample will be connected to its nearest neighbor if that connection does not imply that an occupied area is traversed. Once the goal node is connected to the tree, the search finishes and the robot can start moving in order to reach the goal. RRT is a probabilistically complete and efficient path planning method. However, it might be slow for large environments and other approaches such as RRT* [88] could be implemented.

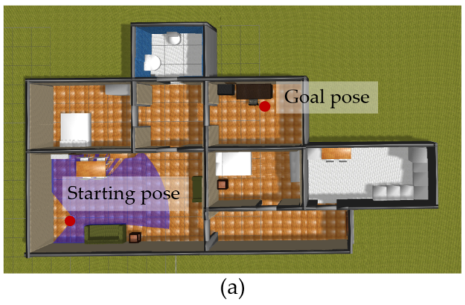
Elastic bands [130] is used with both path planning algorithms in order to soften and optimize the trajectory. Elastic bands consist in building a safe area around the path where the robot is allowed to move. Heuristics are defined to maintain a soft trajectory but keeping the robot as close as possible to the center line of the band. Safe trajectories are guarantied since the band is constrained to the free area around the path. The movement of the robot inside the elastic band is controlled separately regarding its linear and angular speeds. Angular velocity is commanded through a proportional controller based on the angular difference between the current angle of the robot and the current direction of the band. In addition, it takes into account the distance from the robot to the closest point of the line that passes through the center of the band in order to keep the robot as closer as possible to it. Linear velocity is bounded between a minimum and maximum value. Linear velocity can only be lower to the minimum value when the robot is approaching the goal and stopping. Otherwise, the linear velocity will range between its maximum and minimum depending on the current band width and the angular velocity. When crossing narrow areas (small band width) the robot will move slower. In addition, the linear velocity will be decreased according to the angular velocity in order to avoid sharp turns and problems in the stability of the robot.

This hybrid path planning approach is designed to work in static environments as we only perform global path planning methods that rely on the information contained on the traversability graph and the 3D submap. If the traversability graph and the 3D submaps are updated according to the changes of the environment, then the path planning could deal with changes and non-static environments. We have also been working on adapting the local path planning strategy to unmapped changes of the environment [1]. In this way we can modify the elastic band if an unmapped obstacle is detected inside the safe area in order to avoid the collision with that obstacle.

7.3.3 Experimental Evaluation

We are providing a comparison of our method with path planning using a global 3D map in order to demonstrate the advantages of having a hybrid path planning method running on top of a hybrid map. Same starting and goal poses are requested for both representation, using RRT in the global 3D map and the RRT-Dijkstra hybrid path planning method in the hybrid map. In the case of the global map, samples to perform RRT path planning are spread in the 3D map of the environment without taking into

[1]Final Degree Project (TFG) of Ivan Vivares Jimenez, *"Planificador local para robot móvil"*.



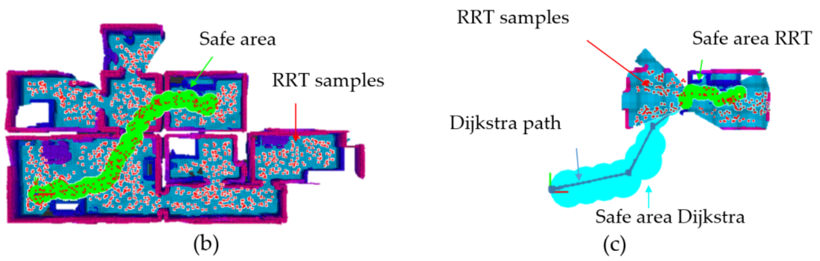


Figure 7.16 Path planning between starting and goal positions(a); the path for global 3D map and hybrid map are shown in (b) and (c). Red is used for samples, green for RRT path safe area and blue for Dijkstra path safe area.

account submaps. In the case of the hybrid map, we first use Dijkstra and samples are generated only in the 3D submap corresponding to the goal room. Figure 7.16 (a) shows the starting and goal positions for an example path. Figure 7.16 (b) shows the result obtained for RRT in the global 3D map. Random samples (red) are generated over the free area of the whole environment. When a path is found, its safe area is built (green) by obtaining the free area around the selected samples. Figure 7.16 (c) shows the result for the hybrid path planning method. Planning starts with Dijkstra algorithm in the traversability graph towards the door of the goal room. A safe area for this topological path is included (blue). Afterwards, RRT is performed in the submap of the goal room. Random samples (red) are generated and a safe area for the metric path (green) is calculated for the samples that connect the door of the goal room to the goal position.

Both path planning methods have been compared according to the planning time (a solution is found), the number of random samples required for the solution, the time to reach the goal and the relative traveled distance (distance traveled compared to Euclidean distance). Average values of paths from the initial position to 7 random positions (one per room) are included in Table 7.5. Planning times for the hybrid path planning are two orders of magnitude smaller than RRT using the global map. The time to reach the goal and the distance traveled have also decreased for the proposed

Table 7.5 Comparison of path planning results for the hybrid map and global 3D map.

Path-planning		t_{plan}[ms]	$n_{samples}$	t_{goal}[s]	$d_{relative}$
Hybrid	Dijkstra	3.08	178.76	242.27	1.58
	RRT	510.7			
	Total	**513.78**			
RRT (Global map)		67.3×10^3	1121.95	480.5	1.93

method. These results uphold that hybrid maps considerably reduce the requirements for path-planning and improve the performance regarding execution time and traveled distance.

The use of hybrid maps and hybrid path planning strategies substantially improves path planning performance in terms of planning and execution time, computational requirements and relative traveled distance. Therefore, we can conclude that hybrid maps allow to perform more efficient path planning strategies.

8

Conclusions and Future Work

This chapter concludes this thesis by summarizing the work and its key contributions. Furthermore, we discuss the results obtained for each chapter and aspects that could be extended or exchanged.

8.1 Conclusions

Together, the approaches proposed in this thesis address a separate yet complementary aspect of robot navigation, starting by mapping and map adaptation and followed by localization and path planning. In GPS-denied environments, such as indoor environments, these tasks are a key requirement to enable autonomous robot operation.

In this thesis, we have presented a hybrid mapping method for static and non-static indoor environments that captures their complexity and diversity. The hybrid map consisted of four different components that represented diverse aspects of the environment in a hierarchical fashion. The first component, the topological map, contained information regarding the structure of the environment identifying rooms and transitions between rooms by doors. The next component, the traversability graph, incorporated the navigable paths in the environment spotting the areas of the paths that corresponded to doors and spaces within rooms. The object-based pose graph included a detailed pose description of the paths and its relation to the objects in the environment. The last component corresponds to a set of 3D metric submaps, one for each room in the environment, that offer a voxel-wise representation of the 3D appearance of each room.

We have completed the study of the conception of the hybrid map by autonomously building the maps, maintaining the representation in the long-term and performing path planning and localization within the given maps. The hybrid map has been autonomously built through a frontier-based exploration algorithm whose cost-utility function was designed to maximize the explored area as fast as possible while minimizing the distance traveled by the robot. Long-term map maintenance was achieved in the object-based pose graph by identifying the objects that were moved or remained in their position during several mapping sessions. In addition, the behaviors in terms of movability for individual objects and object classes were inferred. Nevertheless, we have not solved the issue of long-term maintenance of the whole hybrid map and we intend to address this in the future. Regarding path planning, a strategy that took advantage of the hybrid map was also proposed. The hybrid map allows to combine different path planning strategies for sections of the path enhancing its performance. Finally, we investigated different approaches for localization as the hybrid map offers the possibility to perform topological localization and metric localization in different components of the map. Both localization approaches take advantage of the information contained in the hybrid map. In the case of topological localization, the most probable pose given robot observations is calculated based on the object-based pose graph, the topological map and scene and object information. In the case of metric localization, pose estimation is performed in static and non-static environments using the available metric and semantic information contained in the 3D submaps.

This thesis has contributed especially to the fields of hybrid mapping, map maintenance in non-static environments and path planning and localization within the hybrid map. Regarding hybrid mapping, our main contribution is a novel hybrid map structure that includes structural, relational, 3D an object information that simplifies the construction and abstraction of the map and grants improvements in the navigation capabilities of the robot. In addition, the construction of the map through a new implementation of frontier exploration is more efficient than original frontier exploration and other derived works. Regarding navigation capabilities of the robot, our contributions consist in the idea that path planning and localization benefit from the conception of the hybrid map. A novel path planning strategy is presented that goes beyond standard path planning techniques in terms of planning and execution time and traveled distance. The novelty in topological localization lies in the usage of node, supernode and scene information to increase the probability of estimating the correct node where the robot is. Similarly, the novelty of metric localization is in the inclusion of metric and semantic information to infer the most probable robot pose using a particle filter. In addition, this combination of information makes the system more robust to changes in the appearance of the environment. Regarding long-term operation, we have also proposed a novel map maintenance algorithm that updates the map with regard to the changes in the environment in a probabilistic fashion, obtaining better results than classical binary approaches.

As a general conclusion, we can say that hybrid mapping can produce more efficient maps in terms of memory consumption and the usability of the map is increased. Hybrid maps grant more efficient path planning by using the diversity of information contained in the map and they enable a diversity of localization approaches that benefit from the hybrid information by increasing the precision of robot estimations. In the following, we include specific conclusions for each of the proposed methods.

In Chapters 3 and 4, a novel hybrid mapping method that represents the information as global topological maps and 3D dense submaps is developed. We propose a semantic partitioning of the environment based on the detection of doors that builds a 3D submap for each room without the need of post-processing the information. The resulting hybrid map combines the efficiency of topological maps with the precision of dense representation for mapping, localization and path planning. With regard to mapping, the system outperforms 3D global mapping and other state-of- the-art submapping methods based on fixed-size submaps in terms of memory consumption. We also proposed a probabilistic loop closure method that uses topological, metric and semantic information to validate the loops and solve the correspondences in the map.

In Chapter 5, we have developed an exploration algorithm based on frontier-based exploration and behavior-based strategies that builds the hybrid map of the environment. We proposed using semantic, geometric and topological information of the environment to determine the next best position to visit in indoor environments through a cost–utility function. The novelty of the proposed exploration method is the semantic frontier classification and the cost–utility function for frontier selection. Semantic frontier classification divides the environment into transit areas and free areas. In this work, doors were used as transit areas but other approaches could be considered. For example, in outdoor road environments, transit areas can be road intersections. In this way, if the robot (or autonomous car) is moving on a road, it will map all the nodes along the road path as free area until an intersection is reached where a transit area will be defined. This will isolate the road sections as different connected places. The proposed exploration algorithm could be adapted to other scenarios just including the corresponding transit area detector and some specific constrains (as exploring within the road track limits). Furthermore, through the experimental evaluation, we showed that the proposed method performs better than other state-of-the-art methods in terms of execution time and traveled distance.

In Chapter 6, we have proposed a method for map adaptation through object-based pose graphs for low dynamic indoor environments. This new method calculates and maintains object probability depending on whether an object is seen again or not and capturing whether it is static or movable. As shown in the experimental results, including object probability improves the resulting map. In addition, it provides the robot with a realistic estimation of the movability of objects according to its class gathered from its own experience. The proposed estimation for object class movability outperforms object classification using a binary method. Overall, this work strengthens two ideas: real-world environments have to be treated as dynamic environments consisting of objects that may be more or less movable; and, important information can be obtained from capturing their movability.

Regarding topological localization, in Chapter 7.1, we have presented a hidden Markov model method that localizes a robot within the object-based pose graph and the topological map using scene information. In this work, propagation of scene information is used to improve the final localization. Our main contributions are the abstraction in the hybrid map to localize a robot at different levels and the inclusion of semantic scene and object information in order to improve its localization. As shown in the experiments, the proposed system solves ambiguities among localization results even if errors are intentionally introduced within the observations leading to a more robust estimation.

In Chapter 7.2, we have presented our method for metric localization, which is a novel approach to visual localization in non-static indoor environments. The proposed approach combines metric and semantic information to perform probabilistic pose estimation of a mobile robot. Our method uses information of static and movable objects in the environment as movable elements can contain valuable information for localization. This allows us to successfully estimate robot pose even when the internal representation that the robot has of the environment does not match the current state of the world. We implemented and evaluated our approach on a real indoor environment and provided comparisons to only-metric and only-semantic methods. The results suggest that the combination of metric and semantic information in pose estimation makes the approach more robust as less particles are needed to obtain similar position errors. In addition, it reduces the effect of bad associations due to changes in the environment.

The last method proposed in this thesis is path planning in the hybrid map, in Chapter 7.3. The hybrid path planning method using the hybrid map performs better than RRT on a global 3D map in terms of planning and execution time, computational requirements and relative traveled distance. This is thanks to the combination of different path planning strategies in different sections of the path based on the priority in speed, robustness or precision.

In summary, the work presented in this thesis contributes to the state of the art in robot navigation by pointing out the efficiency that hybrid maps offer during map building and other applications such as localization and path planning. In addition, we highlight the necessity of dealing with the dynamics of indoor environments and the benefits of combining topological, semantic and metric information to the autonomy of a mobile robot.

8.2 Future Work

There are several parts of this thesis that could be extended or exchanged. In this last section, we will summarize the paths open for research.

The first possible future line of work is to perform Simultaneous Localization and Mapping (SLAM) to build the hybrid map. As it is well-known, accurate pose estimation is a requirement for building consistent maps and a consistent map is necessary to accurately estimate a robot pose. Therefore, SLAM is the best approach to build robust maps of indoor environments. In our method, we assume the robot pose is known as we just focus on building the map. In some of the works that constitute this thesis a scan matching algorithm was used to correct the pose given by odometry. However, this method has not been used in the totality of the thesis and, therefore, it was not included. As we simultaneously build the multiple components of the hybrid map, we could combine techniques for topological and metric SLAM and use scan and visual information to improve both, pose estimation and mapping accuracies.

Following with the future work regarding localization, in this thesis, we performed separately topological localization in the topological map and the object-based pose graph and metric localization in a global 3D map. The next step would be to perform concurrent localization in the hybrid map when navigating through it. In this sense, the robot will know where it is in the topological map and the object-based graph (as it does now), in which node of the traversability graph and its exact metric position within the room. This future line of work comprises several tasks. First, as we already mentioned in Chapter 7.1, topological localization is not performed in the object-based pose graph as it has been presented in this thesis but in a previous version of it. Therefore, the first step would be to implement topological localization using the current definition of the object-based pose graph. In addition, current algorithm for metric localization uses a whole 3D map of the environment. This algorithm should be adapted to the submaps of the hybrid map. There are several approaches to adapt the metric estimation to submaps, one could be to use the estimation of the most probable supernode to look for the exact metric position, or, if we want to keep them as independent estimations, the algorithm could load all the submaps until the algorithm converges and then track the robot within the specific submaps.

Metric localization algorithm could be also extended in a variety of ways. In this thesis, the estimation of robot pose was performed through a matching of the metric and semantic information contained in images. For that purpose, image pixels were manually colored according to the semantic class of the objects. Currently, there are algorithms that automatically segment images into semantic regions in terms of objects [116]. It would be a good idea to combine the current metric localization method with any of those semantic segmentation methods to avoid manually labeling images. In addition, metric localization is performed with very dense information as the metric and semantic value for each pixel in the image is compared to each corresponding voxel given the particle position for each of the particles spread in the map. This results in a long estimation process that is far from real-time operation. We would like to study the efficiency of the method and improve it through a downsampling of the information, improving GPU computation or reducing the matching area in the map. Finally, we would like to include the semantic and metric information used in the estimation in the map representation. For example, if the robot is moving in a living room and a chair has been moved to an area that was empty before, we want to add this change in the voxels of the 3D map. As a result, those voxels will contain the semantic label -1 (empty) and 1 (chair). Whenever this voxel is compared again, it will provide two matching possibilities. This is a complex map adaptation method that requires to be treated carefully and with a solid mathematical foundation as we only want to add this information when the localization is precise enough, the semantic class for that voxel is probable enough and the certainty that the selected voxel is the matching one is high enough.

Long-term operation is other future line of the presented work. As we have shown in Chapter 6, map adaptation is a requirement for robust autonomous operation in real-world environments. In this thesis, we have just performed map adaptation in one component of the hybrid map, the object-based pose graph. An interesting future work will be to extend the adaptation to the other components of the map, or at least, as we discussed in Chapter 6, to the traversability graph and the 3D submaps.

Long-term autonomy will also require a more robust approach that can handle dynamic elements also while mapping (high dynamics). In this thesis, we assumed the environment was static while mapping and changes only occurred between mapping sessions. In this sense, if dynamic objects appear while mapping, they will be added to the map and result in inconsistencies in the representation. There are several works that have approached mapping with dynamic objects, such as [17, 124]. Our mapping system will highly benefit from the integration with one of the aforementioned systems.

Up to now, the semantic information contained in the map refers to scenes, objects and doors. In the future, we find appealing to include more semantic information in the map. Similarly to [85], the affordances and possible actions to perform in the environment could be included. In addition, to enhance manipulation tasks, the representation could contain approaching poses for actions to interact with objects.

Finally, only indoor environments populated by humans have been considered in this thesis. An interesting future line of work would be to translate all this knowledge to other environments and tasks, such as road networks and autonomous driving. The exploration method would need some adjustments but the conception of transit area and free area would work similarly enabling the construction of the traversability graph within the road and the topological map and the 3D submaps for each road section.

Bibliography

[1] Alvarez, J. K., Abeywardena, D., Shi, L., and Kodagoda, S. (2015). Using hidden markov models to improve floor level localisation. In *Australasian Conference on Robotics and Automation, ACRA*.

[2] Amigoni, F. (2008). Experimental evaluation of some exploration strategies for mobile robots. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 2818–2823. IEEE.

[3] Amigoni, F., Caglioti, V., and Galtarossa, U. (2004). A mobile robot mapping system with an information-based exploration strategy. In *Intl. Conf. on Informatics in Control, Automation and Robotics (ICINCO-RA)*, pages 71–78.

[4] Amigoni, F. and Gallo, A. (2005). A multi-objective exploration strategy for mobile robots. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3850–3855. IEEE.

[5] Andreasson, H. and Duckett, T. (2004). Topological localization for mobile robots using omni-directional vision and local features. *IFAC Proceedings Volumes*, 37(8):36–41.

[6] Arroyo, R., Alcantarilla, P. F., Bergasa, L. M., and Romera, E. (2016). Fusion and binarization of CNN features for robust topological localization across seasons. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4656–4663. IEEE.

[7] Arroyo, R., Alcantarilla, P. F., Bergasa, L. M., and Romera, E. (2018). Are you able to perform a life-long visual topological localization? *Autonomous Robots*, 42(3):665–685.

[8] Arvanitakis, I., Giannousakis, K., and Tzes, A. (2016). Mobile robot navigation in unknown environment based on exploration principles. In *IEEE Conf. on Control Applications (CCA)*, pages 493–498. IEEE.

[9] Arvanitakis, I. and Tzes, A. (2017). Collaborative mapping and navigation for a mobile robot swarm. In *Mediterranean Conf. on Control and Automation (MED)*, pages 696–700. IEEE.

[10] Bacca, B., Salvi, J., and Cufí, X. (2011). Appearance-based mapping and localization for mobile robots using a feature stability histogram. *Journal on Robotics and Autonomous Systems (RAS)*, 59(10):840–857.

[11] Badino, H., Huber, D., and Kanade, T. (2011). Visual topometric localization. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 794–799. IEEE.

[12] Badino, H., Huber, D., and Kanade, T. (2012). Real-time topometric localization. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 1635–1642. IEEE.

[13] Banerjee, N., Lisin, D., Briggs, J., Llofriu, M., and Munich, M. E. (2019). Lifelong mapping using adaptive local maps. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–8.

[14] Basilico, N. and Amigoni, F. (2011). Exploration strategies based on multi-criteria decision making for searching environments in rescue operations. *Autonomous Robots*, 31(4):401.

[15] Bavle, H., Manthe, S., de la Puente, P., Rodriguez-Ramos, A., Sampedro, C., and Campoy, P. (2018). Stereo visual odometry and semantics based localization of aerial robots in indoor environments. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1018–1023. IEEE.

[16] Beeson, P., Modayil, J., and Kuipers, B. (2010). Factoring the mapping problem: Mobile robot map-building in the hybrid spatial semantic hierarchy. *Intl. Journal of Robotics Research (IJRR)*, 29(4):428–459.

[17] Bescos, B., Fácil, J. M., Civera, J., and Neira, J. (2018). DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters (RA-L)*, 3(4):4076–4083.

[18] Biber, P. and Duckett, T. (2005). Dynamic maps for long-term operation of mobile service robots. In *Robotics: science and systems*, pages 17–24.

[19] Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., and Siegwart, R. (2018). Receding horizon path planning for 3D exploration and surface inspection. *Autonomous Robots*, 42(2):291–306.

[20] Blaer, P. and Allen, P. (2002). Topological mobile robot localization using fast vision techniques. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, volume 1, pages 1031–1036. IEEE.

[21] Blanco, J.-L., Fernández-Madrigal, J.-A., and Gonzalez, J. (2008). Toward a unified Bayesian approach to hybrid metric–topological SLAM. *IEEE Trans. on Robotics (TRO)*, 24(2):259–270.

[22] Blanco, J.-L., González, J., and Fernández-Madrigal, J.-A. (2009). Subjective local maps for hybrid metric-topological slam. *Journal on Robotics and Autonomous Systems (RAS)*, 57(1):64–74.

[23] Blochliger, F., Fehr, M., Dymczyk, M., Schneider, T., and Siegwart, R. (2018). Topomap: Topological mapping and navigation based on visual SLAM maps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3818–3825. IEEE.

[24] Bogue, R. (2016). Growth in e-commerce boosts innovation in the warehouse robot market. *Industrial Robot: An International Journal.*

[25] Boniardi, F., Caselitz, T., Kümmerle, R., and Burgard, W. (2019). A pose graph-based localization system for long-term navigation in CAD floor plans. *Journal on Robotics and Autonomous Systems (RAS)*, 112:84–97.

[26] Bore, N., Ekekrantz, J., Jensfelt, P., and Folkesson, J. (2018). Detection and tracking of general movable objects in large three-dimensional maps. *IEEE Trans. on Robotics (TRO)*, 35(1):231–247.

[27] Bosse, M., Newman, P., Leonard, J., Soika, M., Feiten, W., and Teller, S. (2003). An Atlas framework for scalable mapping. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, volume 2, pages 1899–1906. IEEE.

[28] Bosse, M., Newman, P., Leonard, J., and Teller, S. (2004). Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework. *Intl. Journal of Robotics Research (IJRR)*, 23(12):1113–1139.

[29] Bradley, D. M., Patel, R., Vandapel, N., and Thayer, S. M. (2005). Real-time image-based topological localization in large outdoor environments. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3670–3677. IEEE.

[30] Bulata, H. and Devy, M. (1996). Incremental construction of a landmark-based and topological model of indoor environments by a mobile robot. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, volume 2, pages 1054–1060. IEEE.

[31] Bürki, M., Dymczyk, M., Gilitschenski, I., Cadena, C., Siegwart, R., and Nieto, J. (2018). Map management for efficient long-term visual localization in outdoor environments. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 682–688. IEEE.

[32] Buschka, P. and Saffiotti, A. (2004). Some notes on the use of hybrid maps for mobile robots. In *Proc. of Int. Conf. on Intelligent Autonomous Systems (IAS)*, pages 547–556.

[33] Carrillo, H., Dames, P., Kumar, V., and Castellanos, J. A. (2015). Autonomous robotic exploration using occupancy grid maps and graph slam based on shannon and rényi entropy. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 487–494. IEEE.

[34] Carrillo, H., Dames, P., Kumar, V., and Castellanos, J. A. (2018). Autonomous robotic exploration using a utility function based on Rényi's general theory of entropy. *Autonomous Robots*, 42(2):235–256.

[35] Cassandra, A. R., Kaelbling, L. P., and Kurien, J. A. (1996). Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, volume 2, pages 963–972. IEEE.

[36] Chaari, I., Koubâa, A., Trigui, S., Bennaceur, H., Ammar, A., and Al-Shalfan, K. (2014). Smartpath: An efficient hybrid aco-ga algorithm for solving the global path planning problem of mobile robots. *Intl. Journal of Advanced Robotic Systems*, 11(7):94.

[37] Chebrolu, N., Lottes, P., Läbe, T., and Stachniss, C. (2019). Robot localization based on aerial images for precision agriculture tasks in crop fields. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 1787–1793. IEEE.

[38] Chen, G., Dong, W., Sheng, X., and Zhu, X. (2019a). Generating spatial semantic representations for indoor global mapping. In *IEEE Intl. Conf. on Real-time Computing and Robotics (RCAR)*, pages 19–23. IEEE.

[39] Chen, X., Milioto, A., Palazzolo, E., Giguère, P., Behley, J., and Stachniss, C. (2019b). SuMa++: Efficient LiDAR-based Semantic SLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

[40] Chen, Z., Zhang, Y., Zhang, Y., Nie, Y., Tang, J., and Zhu, S. (2019c). A hybrid path planning algorithm for unmanned surface vehicles in complex environment with dynamic obstacles. *IEEE Access*, 7:126439–126449.

[41] Cheng, H., Chen, H., and Liu, Y. (2014). Topological indoor localization and navigation for autonomous mobile robot. *IEEE Trans. on Automation Science and Engineering*, 12(2):729–738.

[42] Chui, C. K. and Chen, G. (2017). *Kalman filtering.* Springer, Berlin, Germany.

[43] Churchill, W. and Newman, P. (2013). Experience-based navigation for long-term localisation. *Intl. Journal of Robotics Research (IJRR)*, 32(14):1645–1661.

[44] D'Andrea, R. (2012). Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead. *IEEE Transactions on Automation Science and Engineering*, 9(4):638–639.

[45] Dayoub, F. and Duckett, T. (2008). An adaptive appearance-based map for long-term topological localization of mobile robots. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3364–3369. IEEE.

[46] Dayoub, F., Morris, T., Upcroft, B., and Corke, P. (2013). Vision-only autonomous navigation using topometric maps. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1923–1929. IEEE.

[47] de Oliveira, G. C. R., de Carvalho, K. B., and Brandão, A. S. (2019). A hybrid path-planning strategy for mobile robots with limited sensor capabilities. *Sensors*, 19(5):1049.

[48] Dellaert, F., Fox, D., Burgard, W., and Thrun, S. (1999). Monte Carlo localization for mobile robots. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, volume 2, pages 1322–1328. IEEE.

[49] Derner, E., Gomez, C., Hernandez, A., Barber, R., and Babuška, R. (2019). Towards life-long autonomy of mobile robots through feature-based change detection. In *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, pages 1–6. IEEE.

[50] Doan, A.-D., Latif, Y., Chin, T.-J., and Reid, I. (2020). HM4: Hidden Markov model with memory management for visual place recognition. *IEEE Robotics and Automation Letters (RA-L)*, 6(1):167–174.

[51] Du, Z., Qu, D., Xu, F., and Xu, D. (2007). A hybrid approach for mobile robot path planning in dynamic environments. In *IEEE Intl. Conf. on Robotics and Biomimetics (ROBIO)*, pages 1058–1063. IEEE.

[52] Duckett, T., Marsland, S., and Shapiro, J. (2002). Fast, on-line learning of globally consistent maps. *Autonomous Robots*, 12(3):287–300.

[53] Duckett, T. and Saffiotti, A. (2000). Building globally consistent gridmaps from topologies. *IFAC Proceedings Volumes*, 33(27):405–410.

[54] Dudek, G., Jenkin, M., Milios, E., and Wilkes, D. (1991). Robotic exploration as graph construction. *IEEE Trans. on Robotics and Automation*, 7(6):859–865.

[55] Edlinger, T. and von Puttkamer, E. (1994). Exploration of an indoor-environment by an autonomous mobile robot. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, volume 2, pages 1278–1284. IEEE.

[56] Egido, V., Barber, R., and Salichs, M. (2002). Corridor exploration in the EDN navigation system. *IFAC Proceedings Volumes*, 35(1):475–480.

[57] Estrada, C., Neira, J., and Tardós, J. D. (2005). Hierarchical SLAM: Real-time accurate mapping of large environments. *IEEE Trans. on Robotics (TRO)*, 21(4):588–596.

[58] Fehr, M., Furrer, F., Dryanovski, I., Sturm, J., Gilitschenski, I., Siegwart, R., and Cadena, C. (2017). TSDF-based change detection for consistent long-term dense reconstruction and dynamic object discovery. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 5237–5244. IEEE.

[59] Fermin-Leon, L., Neira, J., and Castellanos, J. A. (2017a). Incremental contour-based topological segmentation for robot exploration. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 2554–2561. IEEE.

[60] Fermin-Leon, L., Neira, J., and Castellanos, J. A. (2017b). TIGRE: Topological graph based robotic exploration. In *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, pages 1–6. IEEE.

[61] Fernández-Madrigal, J.-A., Galindo, C., and González, J. (2004). Assistive navigation of a robotic wheelchair using a multihierarchical model of the environment. *Integrated Computer-Aided Engineering*, 11(4):309–322.

[62] Fernández-Moral, E., Arévalo, V., and González-Jiménez, J. (2015). Hybrid metric-topological mapping for large scale monocular slam. In *Informatics in Control, Automation and Robotics*, pages 217–232. Springer.

[63] Forster, C., Sabatta, D., Siegwart, R., and Scaramuzza, D. (2013). RFID-based hybrid metric-topological SLAM for GPS-denied environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 5228–5234. IEEE.

[64] Foskey, M., Garber, M., Lin, M. C., and Manocha, D. (2001). A Voronoi-based hybrid motion planner. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, volume 1, pages 55–60. IEEE.

[65] Fox, D., Burgard, W., and Thrun, S. (1999). Markov localization for mobile robots in dynamic environments. *Journal of artificial intelligence research*, 11:391–427.

[66] Galindo, C., Saffiotti, A., Coradeschi, S., Buschka, P., Fernandez-Madrigal, J.-A., and González, J. (2005). Multi-hierarchical semantic maps for mobile robotics. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2278–2283. IEEE.

[67] Garrote, L., Premebida, C., Silva, D., and Nunes, U. J. (2018). Hmaps-hybrid height-voxel maps for environment representation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1197–1203. IEEE.

[68] Ginés, J., Martín, F., Matellán, V., Lera, F. J., and Balsa, J. (2017). Dynamics maps for long-term autonomy. In *IEEE Intl. Conf. on Autonomous Robot Systems and Competitions (ICARSC)*, pages 85–90. IEEE.

[69] Gomez, C., Hernandez, A. C., Moreno, L., and Barber, R. (2018). Qualitative geometrical uncertainty in a topological robot localization system. In *Intl. Conf. on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*, pages 183–188. IEEE.

[70] González-Banos, H. H. and Latombe, J.-C. (2002). Navigation strategies for exploring indoor environments. *Intl. Journal of Robotics Research (IJRR)*, 21(10-11):829–848.

[71] Gross, J. L. and Yellen, J. (2005). *Graph theory and its applications*. Chapman and Hall/CRC.

[72] Guizzo, E. (2019). Your next salad could be grown by a robot. *IEEE Spectrum*, 57(1):34–35.

[73] Han, S. and Xi, Z. (2020). Dynamic scene semantics SLAM based on semantic segmentation. *IEEE Access*, 8:43563–43570.

[74] Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. on Systems Science and Cybernetics*, 4(2):100–107.

[75] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[76] Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D. (2012). RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *Intl. Journal of Robotics Research (IJRR)*, 31(5):647–663.

[77] Hernandez, A. C., Gomez, C., Barber, R., and Mozos, O. M. (2018). Object-based probabilistic place recognition for indoor human environments. In *Intl. Conf. on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*, pages 177–182. IEEE.

[78] Ho, B.-J., Sodhi, P., Teixeira, P., Hsiao, M., Kusnur, T., and Kaess, M. (2018). Virtual occupancy grid map for submap-based pose graph SLAM and planning in 3D environments. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2175–2182. IEEE.

[79] Holloway, L. (2007). Subjecting cows to robots: farming technologies and the making of animal subjects. *Environment and Planning D: Society and Space*, 25(6):1041–1060.

[80] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206.

[81] Jadidi, M. G., Miro, J. V., and Dissanayake, G. (2018). Gaussian processes autonomous mapping and exploration for range-sensing mobile robots. *Autonomous Robots*, 42(2):273–290.

[82] Jefferies, M. E. and Yeap, W.-K. (2008). *Robotics and cognitive approaches to spatial mapping.* Springer.

[83] Jia, S., Shen, H., Li, X., Cui, W., and Wang, K. (2012). Autonomous robot exploration based on hybrid environment model. In *IEEE Intl. Conf. on Information and Automation*, pages 19–24. IEEE.

[84] Jia, S., Zhao, X., Li, Y., and Wang, K. (2013). A hierarchical approach to region division for mobile robot based on spectral cluster. In *IEEE Intl. Conf. on Robotics and Biomimetics (ROBIO)*, pages 1371–1376. IEEE.

[85] Johnson, C. (2018). *Topological mapping and navigation in real-world environments.* PhD thesis.

[86] Jones, J. L. (2006). Robots at the tipping point: the road to irobot roomba. *IEEE Robotics & Automation Magazine*, 13(1):76–78.

[87] Juliá, M., Gil, A., and Reinoso, O. (2012). A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33(4):427–444.

[88] Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *Intl. Journal of Robotics Research (IJRR)*, 30(7):846–894.

[89] Katsev, M., Yershova, A., Tovar, B., Ghrist, R., and LaValle, S. M. (2011). Mapping and pursuit-evasion strategies for a simple wall-following robot. *IEEE Trans. on Robotics (TRO)*, 27(1):113–128.

[90] Konolige, K., Marder-Eppstein, E., and Marthi, B. (2011). Navigation in hybrid metric-topological maps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3041–3047. IEEE.

[91] Krajník, T., Fentanes, J. P., Mozos, O. M., Duckett, T., Ekekrantz, J., and Hanheide, M. (2014). Long-term topological localisation for service robots in dynamic environments using spectral maps. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4537–4542. IEEE.

[92] Krajník, T., Fentanes, J. P., Santos, J. M., and Duckett, T. (2017). Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments. *IEEE Trans. on Robotics (TRO)*, 33(4):964–977.

[93] Kuipers, B. (2000). The spatial semantic hierarchy. *Artificial Intelligence*, 119(1-2):191–233.

[94] Kuipers, B. and Byun, Y.-T. (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal on Robotics and Autonomous Systems (RAS)*, 8(1-2):47–63.

[95] Kuipers, B., Modayil, J., Beeson, P., MacMahon, M., and Savelli, F. (2004). Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, volume 5, pages 4845–4851. IEEE.

[96] Kuipers, B. J. and Levitt, T. S. (1988). Navigation and mapping in large scale space. *AI Magazine*, 9(2):25–25.

[97] Latif, Y., Cadena, C., and Neira, J. (2013). Robust loop closing over time for pose graph SLAM. *Intl. Journal of Robotics Research (IJRR)*, 32(14):1611–1626.

[98] LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning.

[99] Lázaro, M. T., Capobianco, R., and Grisetti, G. (2018). Efficient long-term mapping in dynamic environments. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 153–160. IEEE.

[100] Lehner, H., Schuster, M. J., Bodenmüller, T., and Kriegel, S. (2017). Exploration with active loop closing: A trade-off between exploration efficiency and map quality. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 6191–6198. IEEE.

[101] Li, G., Chou, W., and Yin, F. (2018). Multi-robot coordinated exploration of indoor environments using semantic information. *Science China Information Sciences*, 61(7):79201.

[102] Li, Q., Zhu, J., Liu, T., Garibaldi, J., Li, Q., and Qiu, G. (2017). Visual landmark sequence-based indoor localization. In *Proc. of Workshop on Artificial Intelligence and Deep Learning for Geographic Knowledge Discovery*, pages 14–23.

[103] Li, S. and Lee, D. (2017). RGB-D SLAM in dynamic environments using static point weighting. *IEEE Robotics and Automation Letters (RA-L)*, 2(4):2263–2270.

[104] Likhachev, M. and Ferguson, D. (2009). Planning long dynamically feasible maneuvers for autonomous vehicles. *Intl. Journal of Robotics Research (IJRR)*, 28(8):933–945.

[105] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 740–755. Springer.

[106] Lindemann, R. A., Bickler, D. B., Harrington, B. D., Ortiz, G. M., and Voothees, C. J. (2006). Mars exploration rover mobility development. *IEEE Robotics & Automation Magazine*, 13(2):19–26.

[107] Liu, Q., Li, R., Hu, H., and Gu, D. (2020). Indoor topological localization based on a novel Deep Learning technique. *Cognitive Computation*, pages 1–14.

[108] Lopez-Perez, J. J., Hernandez-Belmonte, U. H., Ramirez-Paredes, J.-P., Contreras-Cruz, M. A., and Ayala-Ramirez, V. (2018). Distributed multirobot exploration based on scene partitioning and frontier selection. *Mathematical Problems in Engineering*, 2018.

[109] Luo, R. C. and Chiou, M. (2018). Hierarchical semantic mapping using convolutional neural networks for intelligent service robotics. *IEEE Access*, 6:61287–61294.

[110] Mac, T. T., Copot, C., Tran, D. T., and De Keyser, R. (2017). A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization. *Applied Soft Computing*, 59:68–76.

[111] Makarenko, A., Williams, S. B., Bourgault, F., and Durrant-Whyte, H. F. (2002). An experiment in integrated exploration. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 534–539.

[112] Mason, J. and Marthi, B. (2012). An object-based semantic world model for long-term change detection and semantic querying. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3851–3858. IEEE.

[113] McCormac, J., Clark, R., Bloesch, M., Davison, A., and Leutenegger, S. (2018). Fusion++: Volumetric object-level SLAM. In *Intl. Conf. on 3D vision (3DV)*, pages 32–41. IEEE.

[114] Meyer-Delius, D., Beinhofer, M., and Burgard, W. (2012). Occupancy grid models for robot mapping in changing environments. In *Proc. of the Conference on Advancements of Artificial Intelligence (AAAI)*, pages 2024–2030.

[115] Milford, M. and Wyeth, G. (2012). SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 1643–1649. IEEE.

[116] Milioto, A. and Stachniss, C. (2019). Bonnet: An open-source training and deployment framework for semantic segmentation in robotics using CNNs. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 7094–7100. IEEE.

[117] Modayil, J., Beeson, P., and Kuipers, B. (2004). Using the topological skeleton for scalable global metrical map-building. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, volume 2, pages 1530–1536. IEEE.

[118] Newman, P. and Ho, K. (2005). SLAM-loop closing with visually salient features. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 635–642. IEEE.

[119] Niroui, F., Sprenger, B., and Nejat, G. (2017). Robot exploration in unknown cluttered environments when dealing with uncertainty. In *IEEE Intl. Symposium on Robotics and Intelligent Sensors (IRIS)*, pages 224–229. IEEE.

[120] Nitsche, M., de Cristóforis, P., Kulich, M., and Košnar, K. (2011). Hybrid mapping for autonomous mobile robot exploration. In *Proc. of the IEEE Intl. Conf. on Intelligent Data Acquisition and Advanced Computing Systems*, volume 1, pages 299–304. IEEE.

[121] Oishi, S., Inoue, Y., Miura, J., and Tanaka, S. (2019). SeqSLAM++: View-based robot localization and navigation. *Journal on Robotics and Autonomous Systems (RAS)*, 112:13–21.

[122] Oleynikova, H., Taylor, Z., Fehr, M., Siegwart, R., and Nieto, J. (2017). Voxblox: Building 3D signed distance fields for planning. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE.

[123] Oliveira, G. L., Radwan, N., Burgard, W., and Brox, T. (2020). Topometric localization with deep learning. In *Robotics Research*, pages 505–520. Springer.

[124] Palazzolo, E., Behley, J., Lottes, P., Giguère, P., and Stachniss, C. (2019). ReFusion: 3D reconstruction in dynamic environments for RGB-D cameras exploiting residuals. *arXiv preprint*.

[125] Panchpor, A. A., Shue, S., and Conrad, J. M. (2018). A survey of methods for mobile robot localization and mapping in dynamic indoor environments. In *Conf. on Signal Processing And Communication Engineering Systems (SPACES)*, pages 138–144. IEEE.

[126] Patel, N., Khorrami, F., Krishnamurthy, P., and Tzes, A. (2019). Tightly coupled semantic RGB-D inertial odometry for accurate long-term localization and mapping. In *Proc. of the Int. Conf. on Advanced Robotics (ICAR)*, pages 523–528. IEEE.

[127] Paul, R. and Newman, P. (2010). Fab-map 3d: Topological mapping with spatial and visual appearance. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 2649–2656. IEEE.

[128] Paz, L. M., Tardós, J. D., and Neira, J. (2008). Divide and conquer: EKF SLAM in $o(n)$. *IEEE Trans. on Robotics (TRO)*, 24(5):1107–1120.

[129] Pronobis, A. and Jensfelt, P. (2012). Large-scale semantic mapping and reasoning with heterogeneous modalities. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3515–3522. IEEE.

[130] Quinlan, S. and Khatib, O. (1993). Elastic bands: Connecting path planning and control. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 802–807. IEEE.

[131] Radwan, N., Valada, A., and Burgard, W. (2018). Vlocnet++: Deep multi-task learning for semantic visual localization and odometry. *IEEE Robotics and Automation Letters (RA-L)*, 3(4):4407–4414.

[132] Rosen, D. M., Mason, J., and Leonard, J. J. (2016). Towards lifelong feature-based mapping in semi-static environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 1063–1070. IEEE.

[133] Rünz, M. and Agapito, L. (2017). Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 4471–4478. IEEE.

[134] Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H., and Davison, A. J. (2013). SLAM++: Simultaneous localisation and mapping at the level of objects. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1352–1359.

[135] Santos, L., Santos, F. N., Magalhães, S., Costa, P., and Reis, R. (2019). Path planning approach with the extraction of topological maps from occupancy grid maps in steep slope vineyards. In *IEEE Intl. Conf. on Autonomous Robot Systems and Competitions (ICARSC)*, pages 1–7. IEEE.

[136] Santos, L. C., Aguiar, A. S., Santos, F. N., Valente, A., and Petry, M. (2020). Occupancy grid and topological maps extraction from satellite images for path planning in agricultural robots. *Robotics*, 9(4):77.

[137] Schmidt, D., Luksch, T., Wettach, J., and Berns, K. (2006). Autonomous behavior-based exploration of office environments. In *Intl. Conf. on Informatics in Control, Automation and Robotics (ICINCO-RA)*, pages 235–240.

[138] Schmuck, P., Scherer, S. A., and Zell, A. (2016). Hybrid metric-topological 3D occupancy grid maps for large-scale mapping. *IFAC-PapersOnLine*, 49(15):230–235.

[139] Schönberger, J. L., Pollefeys, M., Geiger, A., and Sattler, T. (2018). Semantic visual localization. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 6896–6906.

[140] Sharma, K. R., Honc, D., Dusek, F., and Kumar, G. (2016). Frontier based multi robot area exploration using prioritized routing. In *Proc. of the Europ. Conf. on Modelling and Simulation (ECMS)*, pages 25–30.

[141] Silver, D., Ferguson, D., Morris, A., and Thayer, S. (2006). Topological exploration of subterranean environments. *Journal of Field Robotics*, 23(6-7):395–415.

[142] Skiena, S. (1990). Dijkstra's algorithm. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley*, pages 225–227.

[143] Smith, A. J. and Hollinger, G. A. (2018). Distributed inference-based multi-robot exploration. *Autonomous Robots*, pages 1–18.

[144] Song, J. and Liu, M. (2013). A Hidden Markov model approach for Voronoi localization. In *IEEE Intl. Conf. on Robotics and Biomimetics (ROBIO)*, pages 462–467. IEEE.

[145] Stachniss, C. (2009). *Robotic mapping and exploration*, volume 55. Springer.

[146] Stachniss, C. and Burgard, W. (2005). Mobile robot mapping and localization in non-static environments. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 1324–1329, Pittsburgh, PA, USA.

[147] Stenborg, E., Toft, C., and Hammarstrand, L. (2018). Long-term visual localization using semantically segmented images. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 6484–6490. IEEE.

[148] Stentz, A. (1997). Optimal and efficient path planning for partially known environments. In *Intelligent unmanned ground vehicles*, pages 203–220. Springer.

[149] Suresh, S., Sodhi, P., Mangelson, J. G., Wettergreen, D., and Kaess, M. (2020). Active SLAM using 3D submap saliency for underwater volumetric exploration. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3132–3138. IEEE.

[150] Tang, L., Wang, Y., Ding, X., Yin, H., Xiong, R., and Huang, S. (2019). Topological local-metric framework for mobile robots navigation: a long term perspective. *Autonomous Robots*, 43(1):197–211.

[151] Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71.

[152] Thrun, S., Buecken, A., Burgard, W., Fox, D., Fröhlinghaus, T., Hennig, D., Hofmann, T., Krell, M., and Schmidt, T. (1996). Map learning and high-speed navigation in RHINO.

[153] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics.* MIT Press.

[154] Thrun, S., Gutmann, J.-S., Fox, D., Burgard, W., and Kuipers, B. (1998). Integrating topological and metric maps for mobile robot navigation: A statistical approach. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 989–995.

[155] Toft, C., Stenborg, E., Hammarstrand, L., Brynte, L., Pollefeys, M., Sattler, T., and Kahl, F. (2018). Semantic match consistency for long-term visual localization. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 383–399.

[156] Tomatis, N., Nourbakhsh, I., and Siegwart, R. (2003). Hybrid simultaneous localization and map building: a natural integration of topological and metric. *Journal on Robotics and Autonomous Systems (RAS)*, 44(1):3–14.

[157] Tully, S., Kantor, G., Choset, H., and Werner, F. (2009). A multi-hypothesis topological SLAM approach for loop closing on edge-ordered graphs. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4943–4948. IEEE.

[158] Ulrich, I. and Nourbakhsh, I. (2000). Appearance-based place recognition for topological localization. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, volume 2, pages 1023–1029. Ieee.

[159] Üzer, F., Korrapati, H., Royer, E., Mezouar, Y., and Lee, S. (2016). Vision-based hybrid map building for mobile robot navigation. In *Intelligent Autonomous Systems 13*, pages 135–146. Springer.

[160] Vega-Heredia, M., Ilyas, M., Ghanta, S., Vengadesh, A., Aisyah, S., and Elara, M. R. (2020). Multi-sensor orientation tracking for a facade-cleaning robot. *Sensors*, 20(5):1483.

[161] Vysotska, O. and Stachniss, C. (2016). Lazy data association for image sequences matching under substantial appearance changes. *IEEE Robotics and Automation Letters (RA-L)*, 1(1):213–220.

[162] Vysotska, O. and Stachniss, C. (2019). Effective visual place recognition using multi-sequence maps. *IEEE Robotics and Automation Letters (RA-L)*, 4:1730–1736.

[163] Walcott-Bryant, A., Kaess, M., Johannsson, H., and Leonard, J. J. (2012). Dynamic pose graph SLAM: Long-term mapping in low dynamic environments. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1871–1878. IEEE.

[164] Wang, H., Wang, Z., Yu, L., Wang, Q., and Liu, C. (2018). A hybrid algorithm for robot path planning. In *IEEE Intl. Conf. on Mechatronics and Automation (ICMA)*, pages 986–990. IEEE.

[165] Wang, J. and Chen, Z. (2018). A novel hybrid map based global path planning method. In *Asia-Pacific Conf. on Intelligent Robot Systems (ACIRS)*, pages 66–70. IEEE.

[166] Wang, J. and Yagi, Y. (2013). Efficient topological localization using global and local feature matching. *Intl. Journal of Advanced Robotic Systems*, 10(3):153.

[167] Wilbers, D., Rumberg, L., and Stachniss, C. (2019). Approximating marginalization with sparse global priors for sliding window SLAM-graphs. In *IEEE Intl. Conf. on Robotic Computing (IRC)*, pages 25–31. IEEE.

[168] Worley, R. and Anderson, S. (2020). Topological robot localization in a pipe network. In *Proc. of UKRAS20 Conf.:"Robots into the real world"*, pages 59–60. EPSRC UK-RAS Network.

[169] Wu, W., Li, C., Zuo, W., Zhang, H., Liu, J., Wen, W., Su, Y., Ren, X., Yan, J., Yu, D., et al. (2019). Lunar farside to be explored by chang'e-4. *Nature Geoscience*, 12(4):222–223.

[170] Xiao, L., Wang, J., Qiu, X., Rong, Z., and Zou, X. (2019). Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment. *Journal on Robotics and Autonomous Systems (RAS)*, 117:1–16.

[171] Xu, B., Li, W., Tzoumanikas, D., Bloesch, M., Davison, A., and Leutenegger, S. (2019). Mid-fusion: Octree-based object-level multi-instance dynamic SLAM. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 5231–5237. IEEE.

[172] Xu, X., Hong, B., and Guan, Y. (2017). Humanoid robot localization based on hybrid map. In *Intl. Conf. on Security, Pattern Analysis and Cybernetics (SPAC)*, pages 509–514. IEEE.

[173] Yamanaka, S. and Morioka, K. (2012). Mobile robot navigation using hybrid simplified map with relationships between places and grid maps. *IFAC Proceedings Volumes*, 45(22):616–621.

[174] Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 146–151. IEEE.

[175] Yang, S. and Scherer, S. (2019). CubeSLAM: Monocular 3-D object SLAM. *IEEE Trans. on Robotics (TRO)*, 35(4):925–938.

[176] Youngblood, G. M., Holder, L. B., and Cook, D. J. (2000). A framework for autonomous mobile robot exploration and map learning through the use of place-centric occupancy grids. In *Proc. of the Machine Learning Workshop on Learning From Spatial Information*, pages 25–27.

[177] Yu, J., Deng, W., Zhao, Z., Wang, X., Xu, J., Wang, L., Sun, Q., and Shen, Z. (2019). A hybrid path planning method for an unmanned cruise ship in water quality sampling. *IEEE Access*, 7:87127–87140.

[178] Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., and Funkhouser, T. (2017). 3DMatch: Learning local geometric descriptors from RGB-D reconstructions. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1802–1811.

[179] Zhang, Q. (2015). *Autonomous indoor exploration and mapping using hybrid metric/topological maps.* PhD thesis.

[180] Zhang, Q., Rekleitis, I., and Dudek, G. (2015). Uncertainty reduction via heuristic search planning on hybrid metric/topological map. In *Conf. on Computer and Robot Vision*, pages 222–229. IEEE.

[181] Zhang, W., Liu, G., and Tian, G. (2019). A coarse to fine indoor visual localization method using environmental semantic information. *IEEE Access*, 7:21963–21970.

[182] Zhong, X., Tian, J., Hu, H., and Peng, X. (2020). Hybrid path planning based on safe A* algorithm and adaptive window approach for mobile robot in large-scale dynamic environment. *Journal of Intelligent & Robotic Systems (JINT)*, pages 1–13.

[183] Zhou, R. and Hansen, E. A. (2006). Breadth-first heuristic search. *Artificial Intelligence*, 170(4-5):385–408.

[184] Zhu, J., Li, Q., Cao, R., Sun, K., Liu, T., Garibaldi, J. M., Li, Q., Liu, B., and Qiu, G. (2019). Indoor topological localization using a visual landmark sequence. *Remote Sensing*, 11(1):73.

[185] Ziparo, V. A., Zaratti, M., Grisetti, G., Bonanni, T. M., Serafin, J., Di Cicco, M., Proesmans, M., Van Gool, L., Vysotska, O., Bogoslavskyi, I., and Stachniss, C. (2013). Exploration and mapping of catacombs with mobile robots. In *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–2. IEEE.

[186] Zivkovic, Z., Bakker, B., and Krose, B. (2006). Hierarchical map building and planning based on graph partitioning. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 803–809. IEEE.